2D-Driven 3D Object Detection in RGB-D Images

Jean Lahoud, Bernard Ghanem
King Abdullah University of Science and Technology (KAUST)
Thuwal, Saudi Arabia
{jean.lahoud,bernard.ghanem}@kaust.edu.sa

Abstract

In this paper, we present a technique that places 3D bounding boxes around objects in an RGB-D scene. Our approach makes best use of the 2D information to quickly reduce the search space in 3D, benefiting from state-of-the-art 2D object detection techniques. We then use the 3D information to orient, place, and score bounding boxes around objects. We independently estimate the orientation for every object, using previous techniques that utilize normal information. Object locations and sizes in 3D are learned using a multilayer perceptron (MLP). In the final step, we refine our detections based on object class relations within a scene. When compared to state-of-the-art detection methods that operate almost entirely in the sparse 3D domain, extensive experiments on the well-known SUN RGB-D dataset [29] show that our proposed method is much faster (4.1s per image) in detecting 3D objects in RGB-D images and performs better (3 mAP higher) than the state-of-the-art method that is 4.7 times slower and comparably to the method that is two orders of magnitude slower. This work hints at the idea that 2D-driven object detection in 3D should be further explored, especially in cases where the 3D input is sparse.

1. Introduction

An important aspect of scene understanding is object detection, which aims to place tight 2D bounding boxes around objects and give them semantic labels. Advances in 2D object detection are motivated by impressive performance in numerous challenges and backed up by challenging and large-scale datasets [27, 20, 2]. The progress in 2D object detection manifested in the development and ubiquity of fast and accurate detection techniques. Since 2D object detection results are constrained to the image frame, more information is needed to relate them to the 3D world. Multiple techniques have attempted to extend 2D detections into 3D using a single image [12, 13, 21], but these techniques need prior knowledge of the scene and do not generalize well. With the emergence of 3D sensors (*e.g.* the Microsoft Kinect), which provide depth along with color information, the task of propagating 2D knowledge into 3D becomes more attainable.

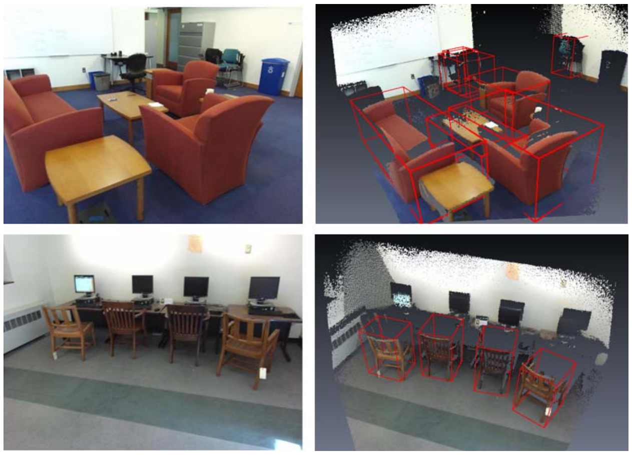

Figure 1. Output of our 2D-driven 3D detection method. Given an RGB image (*left*) and its corresponding depth image, we place 3D bounding boxes around objects of a known class (*right*). We make best use of 2D object detection methods to hone in at possible object locations and place 3D bounding boxes.

The importance of 3D object detection lies in providing better localization that extends the knowledge from the image frame to the real world. This enables interaction between a machine (*e.g.* robot) and its environment. Due to the importance of 3D detection, many techniques replace the 2D bounding box with a 3D one, benefitting from large-scale RGB-D datasets, especially SUN RGB-D [29], which provides 3D bounding box annotations for hundreds of object classes.

One drawback of state-of-the-art detection methods that operate in 3D is their runtime. Despite hardware acceleration (GPU), they tend to be much slower than 2D object detection methods for several reasons. **(i)** One of these reasons is the relative size of the 3D scene compared to its 2D image counterpart. Adding an extra spatial dimension

2380-7504/17 $31.00 © 2017 IEEE
DOI 10.1109/ICCV.2017.495

greatly increases the search space in 3D, and thus slows the search process down. **(ii)** Another reason is the incomplete sparse data available in 3D point clouds generated by a single RGB-D image, which suffers from weak adjacency/contiguity characteristics found in 2D images. **(iii)** The ideal encoding and exploitation of depth information in RGB-D images is still an open challenge. Techniques in the literature have either tried augmenting the color channels with depth, or encoding it into a sparse voxelized 3D scene. Using the depth information as an additional channel aids the detection process, while still benefiting from fast 2D operations, but end results are limited to 2D detections in the form of 2D bounding boxes or 2D object segmentations. Information that can be encoded in 3D include densities, normals, gradients, signed distance functions, among others. Nonetheless, all these 3D voxelization-based techniques suffer from the large amount of missing 3D information, whereby the observable points in a scene only constitute a small fraction of 3D voxels.

In this paper, we propose a 3D object detection method that benefits from the advances in 2D object detection to quickly detect 3D bounding boxes. An output of our method is shown in Figure 1. Instead of altering 2D techniques to accept 3D data, which might be missing or not well-defined, we make use of 2D techniques to restrict the search space for our 3D detections. We then exploit the 3D information to orient, place, and score the bounding box around the desired object. We use previous methods to orient every object independently, and then use the obtained rotation along with point densities in each direction to regress the object extremities. Our final 3D bounding box score is refined using semantic context information.

In addition to the speedup gained from honing into the part of the 3D scene that might contain a particular object, 3D search space reduction also benefits the overall performance of the detector. This reduction makes the 3D search space much more amenable to a 3D method than searching the entire scene from scratch, which slows down search and generates many undesirable false positives. These false positives could confuse a 3D classifier, which is weaker than the 2D classifier because it is trained on sparse (mostly empty) 3D image data.

2. Related Work

There is a rich literature on computer vision techniques that detect objects by placing rectangular boxes around them. We here mention some of the most representative methods that address this problem, namely DPM (deformable parts model) [3] and Selective Search [33] before the ubiquity of deep learning based methods, as well as, representative deep networks for this task including R-CNN [6], Fast R-CNN [5], Faster R-CNN [24], ResNet [10], YOLO [23], and R-FCN [14]. All of these techniques target object detection in the 2D image plane only, and have progressed to become very fast and efficient for this task.

With the emergence of 3D sensors, there have been numerous works that use 3D information to better localize objects. Here, we mention several of these methods [18, 19, 15, 32], which study object detection in the presence of depth information. Other techniques semantically segment images based on RGB and depth, such as [9, 8, 17, 25, 28]. All of these 3D-aware techniques use the additional depth information to better understand the images in 2D, but do not aim to place correct 3D bounding boxes around detected objects.

The method of [30] uses renderings of 3D CAD models from multiple viewpoints to classify all 3D bounding boxes obtained from sliding a window over the whole space. Using CAD models restricts the classes and variety of objects that can be detected, as it is much more difficult to find 3D models of different types and object classes than photograph them. Also, the sliding window strategy is computationally demanding, rendering this technique quite slow. Similar detectors use object segmentation along with pose estimation to represent objects that have corresponding 3D models in a compiled library [7]. Nevertheless, we believe that correct 3D bounding box placement benefits such a task, and model fitting can be performed depending on the availability of these models. When compared to [7], our method does not require 3D CAD models, is less sensitive to 2D detection errors, and improves detection using context information.

Other methods propose 3D boxes and score them according to hand-crafted features. The method proposed in [1] places 3D bounding boxes around objects in the context of autonomous driving. The problem is formulated as inference in an MRF, which generates proposals in 3D and scores them according to hand-crafted features. This method uses stereo imagery as input and targets only the few classes specific to street scenes. For indoor scenes, the method presented in [19] uses 2D segmentation to propose candidate boxes and then classifies them by forming a conditional random field (CRF) that integrates information from different sources. The recent method of [26] proposes a cloud of oriented gradients descriptor, and uses it, along with normals and densities, to classify 3D bounding boxes. This method also uses contextual features to better propose boxes in 3D by exploiting a cascaded classification framework from [11]. This method achieves state-of-the-art performance on SUN-RGBD; nevertheless, computing the features for all 3D cuboid hypotheses is very slow (10-20 minutes per class).

Recent works have also applied ConvNets for 3D object detection. One 3D ConvNet approach is presented in [31], which takes a 3D volumetric scene from the RGB-D image and outputs 3D bounding boxes. The two main modules in this approach are the Region Proposal Network (RPN) and the Object Recognition Network (ORN). The use of object proposals is inspired from 2D object detection techniques. Nevertheless, the two networks do not share layers and computations are done separately. Moreover, the 3D encoding of depth into a truncated signed function, and 3D convolutions are much slower when compared to their 2D counterparts. This ConvNet approach takes about 20s to run on a single RGBD frame. Another recent ConvNet approach [16] presents a transformation network that takes as input the 3D volumetric representation of the scene and aligns it with a known template. 3D object detection is then based on local object features, and on holistic scene features. Although this algorithm is fast at test time (0.5s), training is computationally expensive (one week with 8 GPUs), and it does not generalize to all scene configurations (tested on $\sim$7% of SUN RGB-D testing set) .

Contributions. We propose a fast technique that places bounding boxes around objects using RGB-D data only. Our method does not use CAD models, but places 3D bounding boxes, which makes it easily generalizable to other object classes. By honing in on where particular object instances could be in 3D (using 2D detections), our 3D detector does not need to exhaustively search the whole 3D scene and encounters less false positives that might confuse it. When compared against two state-of-the-art 3D detectors that operate directly in 3D, our method achieves a speedup that does not come at the expense of detection accuracy.

3. Methodology

Given an RGB image and its corresponding depth image, we aim to place 3D bounding boxes around objects of a known class. Our 3D object detection pipeline is composed of four modules (refer to Figure 2 for an overview). In the first module, we use a state-of-the-art 2D object detection method, specifically Faster R-CNN [24], to position 2D bounding boxes around possible objects. Each 2D bounding box extends in 3D to what we call a *frustum*. In the second module and unlike previous methods [31] that assume all objects in the scene share the same orientation, we estimate scene and individual object orientations, where every object has its own orientation. In the third module, we train a Multi-Layer Perceptron to regress 3D object boundaries in each direction, using point densities along oriented directions. In the final module, we refine the detection scores using contextual information that is based on object class co-occurrence and class-to-class distance.

3.1. Slit Detection

The first step in our 3D detection pipeline is to get an initial estimate of the location of objects in 2D. Here, we choose to use the Faster R-CNN model [24] with VGG-16 net to train a detector on the set of object classes found in the 3D dataset (SUN-RGBD). In 2D, the detected object is represented by a 2D window. In 3D, this translates into a 3D extension, which we call a *frustum*. An object's frustum corresponds to the 3D points whose projections onto the image plane are contained within the 2D detection window. As such, the potential object would be present in the region bounded by the planes passing through the camera center and the line segments of the bounding box in the 2D image. A frustum resembles a cone with a rectangular base. In essence, this region provides a much smaller 3D search space than the whole region captured by the RGB-D sensor. In addition to that, each frustum is designated with only the object class that the 2D detector returns.

2D object detection benefits from the continuity of information in the image, and 2D convolutions include RGB information for all the locations targeted. This makes the 2D information more reliable for object detection and classification, when compared to the missing 3D data in the voxelization of the 3D scene. Moreover, a forward pass through the Faster R-CNN runs at least at 5 fps on a GPU, which is two orders of magnitude faster than Deep Sliding Shapes (DSS) [31] that uses 3D convolutions.

Within every frustum, 3D points are spread between the point with the smallest depth and the one with the largest. These points retain all the depth information needed to properly detect the object. When compared to the exhaustive sliding window approach, this is similar to fixating at a specific 2D region instead of searching the whole area looking for all object classes.

3.2. Estimating 3D Object Orientation

Thus far, we have determined the regions that most likely contain an object class. Our next step is to estimate the orientation of the object within this region. Since 3D bounding boxes are of Manhattan structure, object orientations must then be aligned with the best Manhattan frame. This frame would estimate the orientation of the object, since most 3D objects found in indoor scenes can be approximated as Manhattan objects, whose normals are aligned with three main orthogonal directions.

To compute this Manhattan frame, we use the Manhattan Frame Estimation (MFE) technique proposed in [4] to independently estimate the orientation of the object within every frustum. In summary, the rotation $\mathbf{R}$ can be found by solving the following optimization problem

$$\min_{\mathbf{R},\mathbf{X}} \ \frac{1}{2}\|\mathbf{X}-\mathbf{R}\mathbf{N}\|_F^2 + \lambda\|\mathbf{X}\|_{1,1} \tag{1}$$

where $\mathbf{N}$ is the matrix containing the normals at every 3D point, λ is a constant parameter, and $\mathbf{X}$ is a slack variable introduced to make $\mathbf{RN}$ sparse.

Here, we assume that there is only one main object within each frustum. We initially compute the normals for

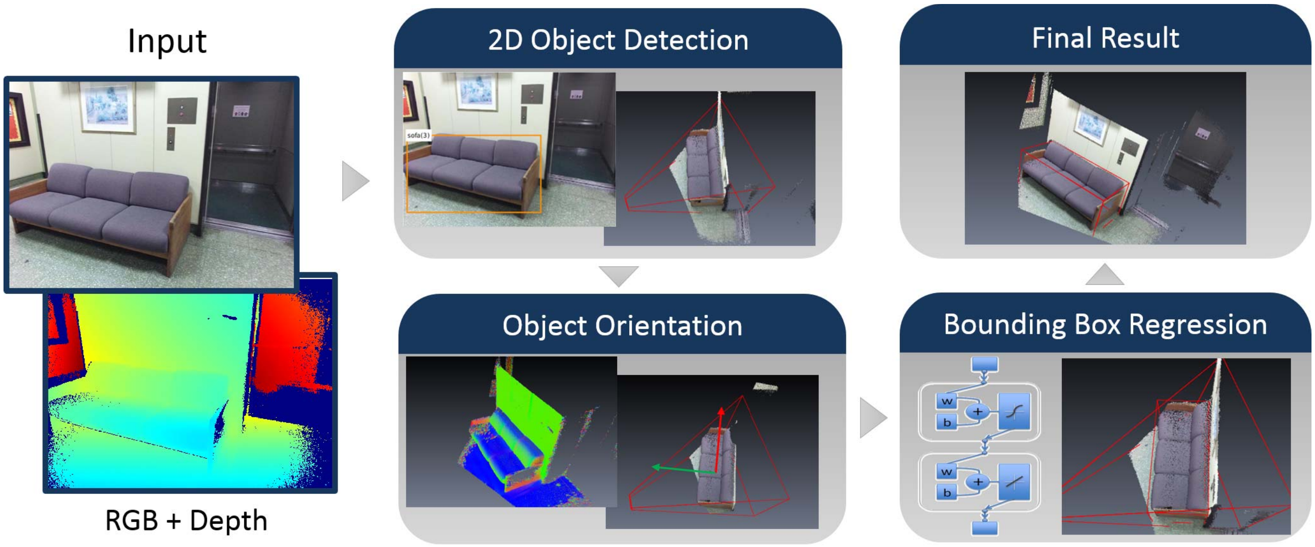

Figure 2. Proposed pipeline. Starting with an RGB image and its corresponding depth image, 2D detection window is used to crop the 3D scene into a frustum. Normals are then used to estimate the object orientation within the frustum. At the final step, an MLP regressor is used to regress the object boundaries based on the histograms of points along x,y, and z directions.

all the 3D points in the image and use MFE to orient the whole scene with respect to the camera. For every frustum, we initialize with the room orientation and use the normals of the points within to estimate the object orientation. In this paper, we modify MFE to restrict the rotation to be around the axis along the normal of the floor (yaw angle only). This restriction is a viable assumption for most of the objects in indoor scenes and aligns with how SUN RGB-D dataset was annotated. For objects of non-Manhattan structure (*e.g.* round objects), many orientations are considered correct. The output of the MFE technique would still be a feasible orientation for object detection.

3.3. Bounding Box Regression

In this step, we need to fit the 3D bounding box that best describes the object being detected. Given the 3D points within the frustum along with the estimated orientation of the object, we place an orthonormal system centered at the centroid of the 3D points and oriented with the estimated orientation. We then construct histograms for the coordinates of the 3D points along every direction. These histograms are then used as input to a multilayer perceptron (MLP) network that learns to regress the boundaries of the bounding box of the object from training data. For every object class, a network with one hidden layer is trained to take as input the coordinate histograms and output the bounding box boundaries of the object along every direction. Histograms describe the density of points along every direction, and high densities correspond to surface locations. We choose a constant bin size to preserve real distances, and choose a constant histogram length to account for all object sizes. An example for the input to the MLP is presented in the **supplementary material**.

The training is done on every direction separately namely, length, width, and height. During testing, the height is known from the ground orientation, and the length and width are specified from the broader distribution of the points along every direction within the frustum. To form the training set, we use the 2D groundtruth windows along with the groundtruth 3D boxes. Since many of the indoor objects are placed on the floor, we use height information from the training set to clip the height of objects close to the floor to start from the floor.

Once the 3D bounding box is obtained, we assign to it a score which is a linear combination of two scores: (1) the initial 2D detection score, and (2) 3D point density score. The 3D point density score is computed by training a linear SVM classifier on the 3D point cloud density of the 3D cuboids for all classes. This simple 3D feature is similar to and inspired by the one used in [26]. Clearly, other 3D features can also be incorporated, but at the expense of added computational cost. We train our classifier using all possible rotations of objects, along with slight variations in object locations.

3.4. Refinement Based on Context Information

Given a set of 3D bounding boxes $\{B_i : i \in \{1, 2, ..., n_b\}\}$ where n_b is the number of boxes, we aim to refine their scores based on contextual information. We associate to every box B_i a label l_i where $l_i \in \{0, 1, ..., n_l\}$.

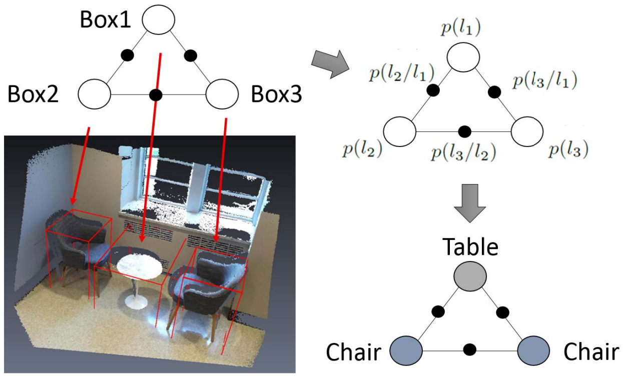

Figure 3. We model the bounding box labels as a set of discrete random variables that can take discrete values. The factor graph is composed of a set of variable nodes (bounding box labels), and a set of factor nodes between every two boxes in a given scene.

Here n_l is the number of object class labels considered, and the zero label corresponds to the background. We assume that the bounding box labels $L = \{l_1, l_2, ..., l_{n_b}\}$ are a set of discrete random variables that has a Gibbs distribution associated with a factor graph G. The factor graph is composed of a set of variable nodes (bounding box labels), and a set of factor nodes P, which we choose to be any combination of 2 bounding boxes. The graph is constructed by assigning all objects in a given scene to one particular node. Our graph model is shown in Figure 3. In this case, we have

$$P_{U,B}(l) \propto exp\left(\sum_{i=1}^{n_b} U_i(l_i) + \sum_{(i,j)\in P} B(l_i, l_j)\right)$$

Here U an B are the unary and binary log potential functions. Our goal is to find the labelling that maximizes the aposteriori (MAP),

$$L = \arg\max_{l} P_{U,B}(l)$$

The above problem can be transformed into a linear program (LP) by introducing the local marginal variables p_u and p_b [34],

$$(p_u, p_b) = \arg\max \sum_{i=1}^{n_b} \mathbb{E}_{p_u}[U_i(l_i)] + \sum_{(i,j)\in P} \mathbb{E}_{p_b}[B(l_i, l_j)]$$

Probability associated with unary term. To model the unary potential p_u, which represents the probability of giving box B_i a label l_i, we train an error-correcting output codes multi-class model using one-versus-one SVM classifiers with cubic polynomial kernels. We append two types of features, geometric features and deep learning features. The geometric features consist of the length, width, height, aspect ratios, and volume. To extract the deep learning features, we run Fast RCNN [5] on the reprojection of the 3D box into the image plane, and use the features from the fully connected layer (FC7). During testing, we transform classification scores to class posterior probabilities and use them as unary probabilities.

Probability associated with binary term. For every pair of 3D boxes, we compute the probability of assigning one box a label l_i given that the other is labelled l_j. We make use of two relations that occur within a 3D scene, class co-occurrence and class spatial distribution. For the co-occurrence probability p_o, we use the number of co-occurences of every combination of two classes in the training set. Given a label l_i, p_o is the ratio between the number of occurrence of l_j and all other occurrences. As for the spatial relation, we use kernel density estimation (KDE) to assign the probability p_d based on the Hausdorff distance between a pair of bounding boxes. Finally, we define the final binary term probability as: $p_b = p_o^{\alpha} p_d^{1-\alpha}$.

To trade-off between the unary and binary terms, we use the softmax operator. To infer the final set of labels, we use the LP-MAP technique of [22]. We then compare the final set of labels to the initial ones and increase the score of the ones that retain their initial labels.

4. Experiments

We evaluate our technique on the SUN RGB-D dataset [29] and compare with two state-of-the-art methods: Deep Sliding Shapes (DSS) [31] and Cloud of Oriented Gradients (COG) [26]. We adopt the 10 categories chosen by COG to train and test on. We also use the dataset modification presented in [31], which provides the floor orientation. We adopt the same evaluation metric as these two methods, *i.e.* we assume that all bounding boxes are aligned with the gravity direction.

Evaluation Metric. We base our evaluation on the conventional 3D volume Intersection over Union (IoU) measure. We consider a detection to be correct, if the volume IoU of the resulting bounding box with the groundtruth box is larger than 0.25. We follow the same evaluation strategy adopted by both DSS and COG. We plot the precision-recall graphs for the 10 classes, and calculate the area under the curve, represented by the Average Precision (AP). Similar to previous methods, we compare to the amodal groundtruth bounding box, which extends beyond the visible range.

Experimental Setup. In our 2D object detector, we follow dataset augmentation convention and add flipped images to the training set. We initialize all convolution layers in the Faster R-CNN networks with a model pre-trained on

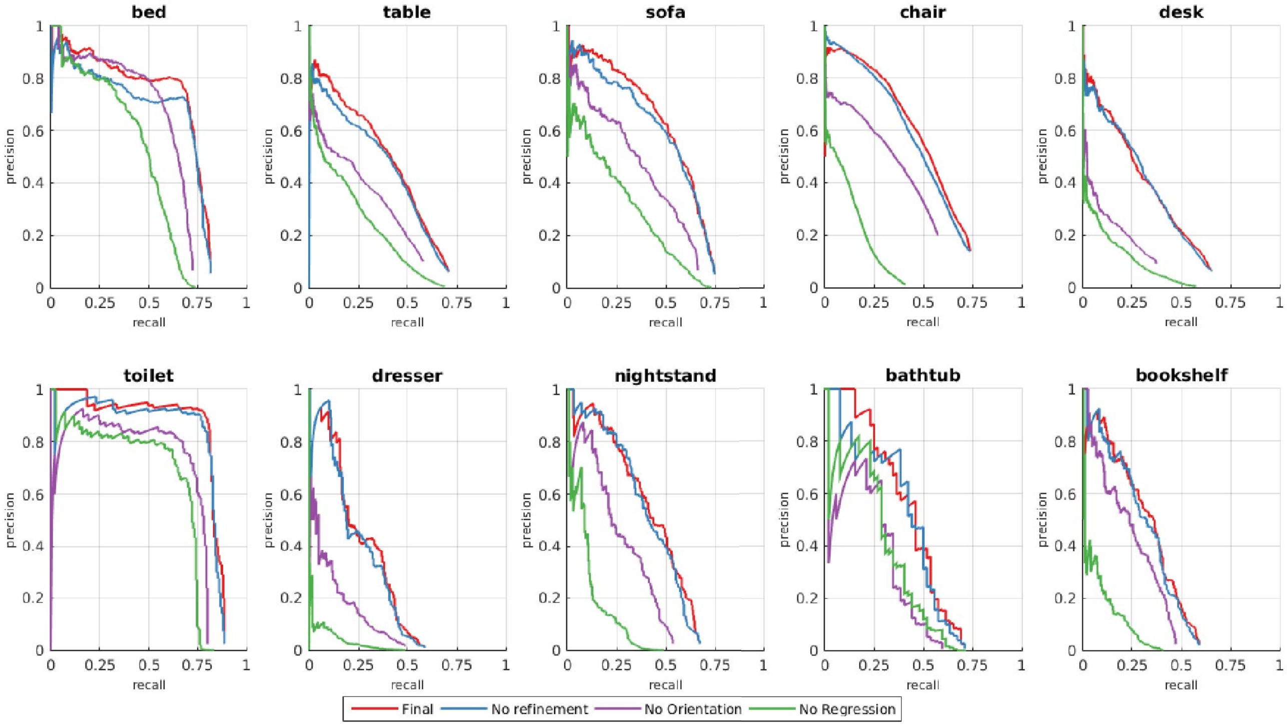

Figure 4. Precision-Recall curve for the proposed 3D object bounding boxes for the classes considered in [26]. We here test the importance of each of our modules; namely, label refinement based on context information, object orientation, and MLP regression.

ImageNet [27]. We also adopt the 4-step alternating training described in [24]. In all our experiments, we consider the 1st percentile of all the 3D point coordinates along the direction that is normal to the floor as the camera height.

After our initial 2D detection, we remove all boxes that overlap by more than 30% with higher scoring boxes, and remove all boxes with very low scores (less than 0.05). To orient the main Manhattan frame, we subsample the normals 10 times, which enables real-time processing for this stage of the pipeline. Once the room orientation is estimated, we compute orientations and regressions of every frustum in parallel. Further details on the implementation of the bounding box regression (Section 3.3) and refinement using contextual relations (Section 3.4) are provided in the **supplementary material**.

Computation Speed. We implement our algorithm in MATLAB, with a 12-core 3.3 GHz CPU and a Titan X GPU. Faster R-CNN training and testing is done using Caffe. The initial object detection takes 0.2 seconds only. When run in parallel, object orientation estimation takes about 1 second for all objects. MLP regression takes about 1.5 seconds for all objects along x, y, and z directions. The context-based object refinement runs in 1.45 seconds, including the time to set up the factor graph and infer the final labels. In summary, our detection method requires a total of 4.15 seconds for every RGB-D image pair.

4.1. Quantitative Comparison

First, we study the importance of every stage in our proposed pipeline. We plot the precision-recall curve for different variants of our method (see Figure 4). **(1)** The first variant is denoted 'no refinement', where we do not perform the last refinement step that incorporates the scene context information. In this case, we use the labels that are output by the 2D object detector along with their corresponding scores. **(2)** We also try to fit boxes to frustums, when all the boxes in all frustums are given the same orientation (room orientation). **(3)** The last variant of our algorithm does not regress object boundaries using the MLP regressor. We replace the regressed box by one that extends up to a percentile of the maximum and minimum coordinate in every direction. Clearly, this cannot handle amodal boxes.

Fixed vs Independent Orientation. We study the importance of correctly orienting 3D bounding boxes. When compared to one fixed orientation, computing the correct orientation for each bounding box increases the final score (Table 1) because of the higher overlap between objects of the same orientation and because the orientation is crucial to fit the correct object boundaries in the MLP regressor.

	🛁	🛏		🪑				🛋		🚽	Runtime	mAP
COG [26]	58.26	63.67	31.80	62.17	45.19	15.47	27.36	51.02	51.29	70.07	10-30min	47.63
DSS [31]	44.2	78.8	11.9	61.2	20.5	6.4	15.4	53.5	50.3	78.9	19.55s	42.1
Ours no regression	29.79	45.22	6.90	10.16	7.29	2.21	10.43	22.96	18.03	59.62	1.3s	21.26
Ours no orientation	25.18	56.20	23.35	31.03	9.20	9.55	25.86	33.95	23.00	65.11	1.8s	30.24
Ours no refinement	40.41	59.31	30.10	46.01	27.26	24.65	40.57	47.66	34.38	77.49	2.7s	42.79
Ours final	43.45	64.48	31.40	48.27	27.93	25.92	41.92	50.39	37.02	80.40	4.15s	45.12

Table 1. Average precision for the 10 classes considered in our experiments.

The Importance of Regression. The importance of regression lies in both localizing the object center and estimating bounding box dimensions. Due to the noisy nature of the 3D data along with the presence of background points, the centroid of the 3D points within the frustum is different than the center of the object. Without regression, the detection score drops significantly (Table 1). This is caused by the incorrect object sizes that directly relate to the detection score. Moreover, since the groundtruth boxes are amodal, meaning that they extend beyond the visible part, regression is needed to extend boxes beyond the visible points.

The Importance of Contextual Information. In the final form of our technique, we incorporate context information to refine the final detection. This refinement increases the mAP score by more than 2% (Table 1). When compared to the ”no refinement” variant in Figure 4, we notice that the refinement enhances the precision at the same recall. On the other hand, it achieves the same maximum recall, since it does not alter the number of 3D boxes nor their locations.

In the second part, we compare against two state-of-the-art methods, DSS and COG (Table 1). We report the runtime and AP results directly from [31] and [26]. We present two versions of our technique: the final version (that includes all the modules in our pipeline) and the no-refinement version (that does not employ context information but is about 35 % faster). The no-refinement version is 7× faster than DSS and is slightly better in terms of accuracy. The final version is 4.7× faster and 3% mAP more accurate than DSS (45.1% mAP vs 42.1% mAP). It is also about two orders of magnitude faster than COG, while still achieving a comparable detection performance.

Unlike DSS, which exploits the color information at late stages of detection, our technique utilizes this information at the beginning of the detection process. We believe that the color information is a crucial part in the 3D object detection, due to the high noise in the 3D information. Our approach mitigates the extensive 3D search with many erroneous boxes arising from the absence of color information. Furthermore, as shown in our experiments, greatly reducing the search space using color information does not come at the expense of detection accuracy.

When compared to COG, our method only uses object-to-object relations and not object-to-scene relations. We do not exhaustively search for 3D bounding boxes, but still manage to achieve a comparable accuracy, while being two orders of magnitude faster. The COG method spends most of the time computing features for all possible 3D bounding boxes locations, sizes, and orientations. Our method hones in on likely object locations, and uses only one orientation. The COG method does not perform much better than our result because of the many object proposals that might confuse the classifier, especially with the sparse 3D data.

4.2. Qualitative Comparison

Now, we show some of our qualitative results. In Figure 5, we overlay the detection results (in *red*) of our method on the 3D point clouds of eight RGB-D images from SUN-RGBD and compare them to the 3D groundtruth bounding boxes shown in *green*. Our method is able to correctly place the bounding box in terms of orientation and extent. We also show false detections from our proposed technique in Figure 6. This includes objects that are not detected in 2D, or objects that are misplaced using the output of the MLP. Misplacement occurs due to background points or incorrect regression. Also, false positives include objects from unseen object classes that look similar to a class seen in training (*e.g.* counter vs. desk and drawer vs. dresser).

5. Conclusion

We propose a fast algorithm for 3D object detection in indoor scenes. We use 2D detections to hone in on the potential 3D locations of particular object classes in 3D (called slit generation and carving), which enables the use of a simple 3D classifier and search mechanism. When compared to two recent state-of-the-art methods, our method is 3 times faster and achieves better (+3% mAP) detection performance than one of the methods, and two orders of magnitude faster than the other while still performing comparably on the challenging and large-scale SUN-RGBD dataset.

Acknowledgments. This work was supported by the King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research.

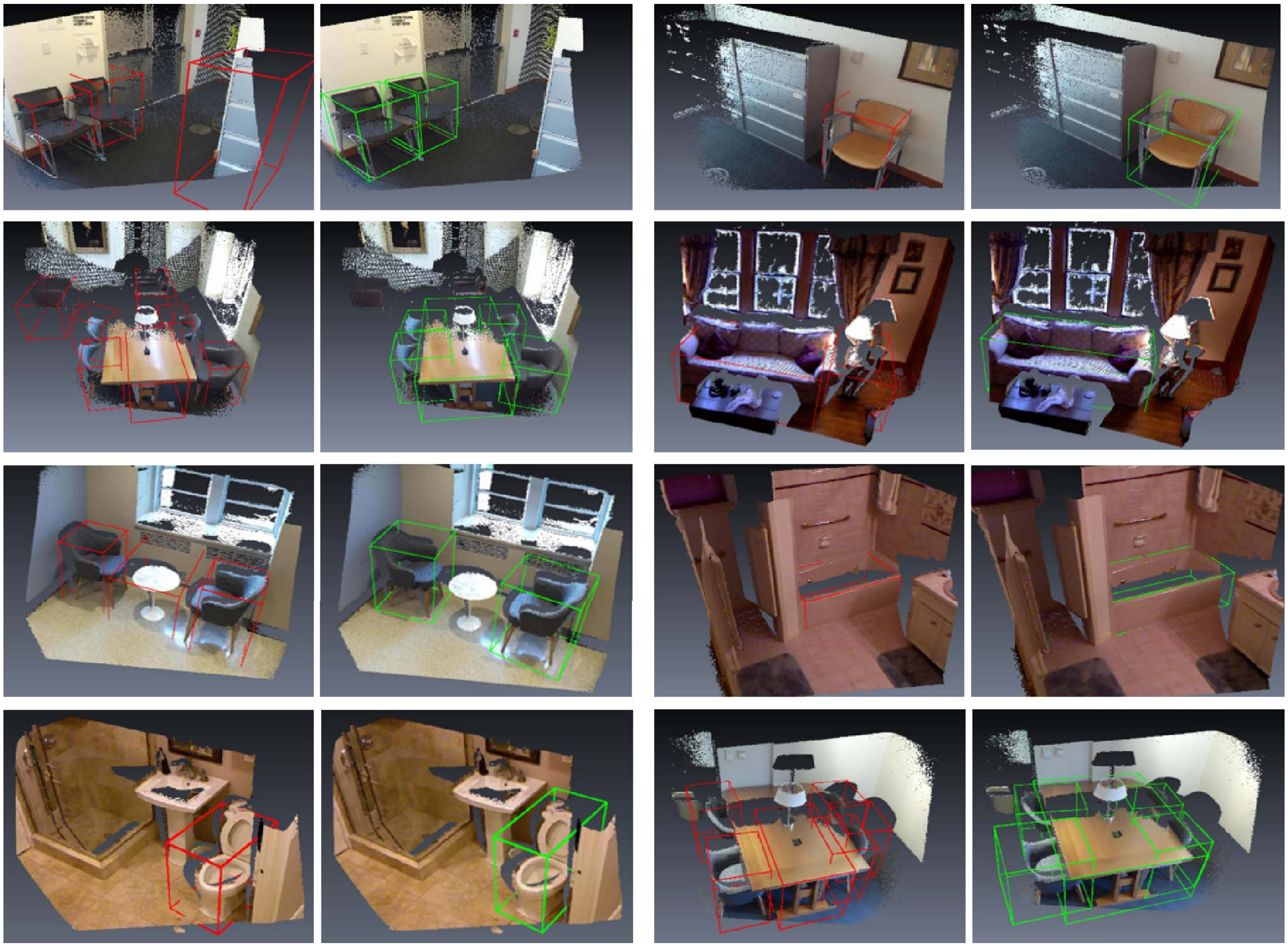

Figure 5. Detections obtained from our method having a score larger than a constant threshold (red). Groundtruth boxes are shown in (green).

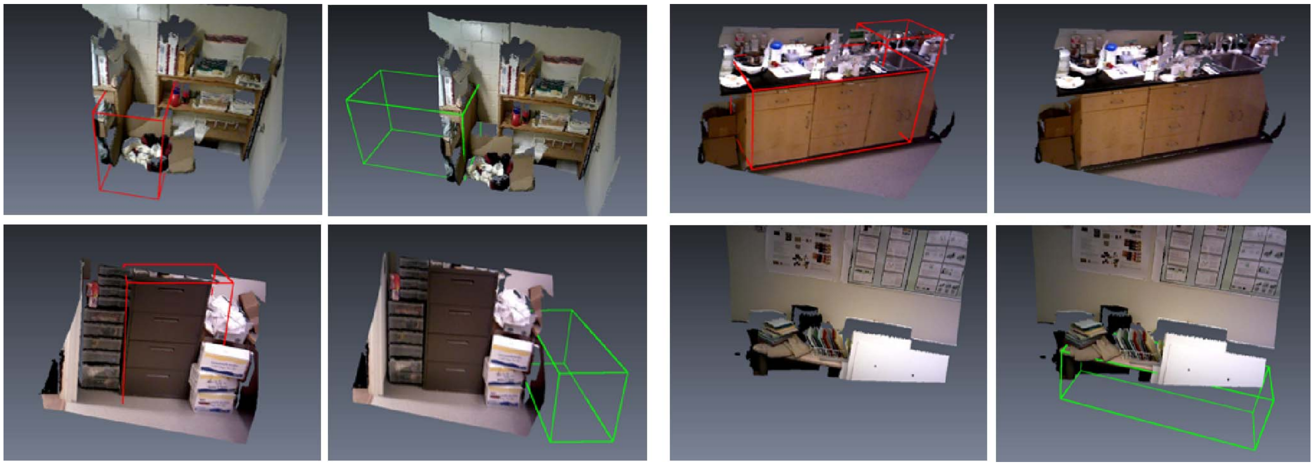

Figure 6. False detections from our proposed technique, including objects that are not detected in 2D, objects that are misplaced using the output of the MLP, and false positives from unseen object classes, but look similar to a class in the training set (counter vs desk, drawer vs dresser).

References

[1] X. Chen, K. Kundu, Y. Zhu, A. G. Berneshawi, H. Ma, S. Fidler, and R. Urtasun. 3d object proposals for accurate object class detection. In *Advances in Neural Information Processing Systems*, pages 424–432, 2015.

[2] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88(2):303–338, 2010.

[3] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 32(9):1627–1645, 2010.

[4] B. Ghanem, A. Thabet, J. C. Niebles, and F. C. Heilbron. Robust manhattan frame estimation from a single rgb-d image. In *CVPR*, pages 3772–3780. IEEE, 2015.

[5] R. Girshick. Fast r-cnn. In *ICCV*, pages 1440–1448, 2015.

[6] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, pages 580–587, 2014.

[7] S. Gupta, P. Arbeláez, R. Girshick, and J. Malik. Aligning 3d models to rgb-d images of cluttered scenes. In *CVPR*, pages 4731–4740, 2015.

[8] S. Gupta, P. Arbelaez, and J. Malik. Perceptual organization and recognition of indoor scenes from rgb-d images. In *CVPR*, pages 564–571, 2013.

[9] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik. Learning rich features from rgb-d images for object detection and segmentation. In *ECCV*, pages 345–360. Springer, 2014.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[11] G. Heitz, S. Gould, A. Saxena, and D. Koller. Cascaded classification models: Combining models for holistic scene understanding. In *Advances in Neural Information Processing Systems*, pages 641–648, 2009.

[12] D. Hoiem, A. A. Efros, and M. Hebert. Geometric context from a single image. In *ICCV 2005*, volume 1, pages 654–661. IEEE, 2005.

[13] D. Hoiem, A. A. Efros, and M. Hebert. Putting objects in perspective. *IJCV*, 80(1):3–15, 2008.

[14] K. H. J. S. Jifeng Dai, Yi Li. R-FCN: Object detection via region-based fully convolutional networks. *arXiv preprint arXiv:1605.06409*, 2016.

[15] B.-s. Kim, S. Xu, and S. Savarese. Accurate localization of 3d objects from rgb-d data using segmentation hypotheses. In *CVPR*, pages 3182–3189, 2013.

[16] Y. Z. M. B. P. Kohli, S. Izadi, and J. Xiao. Deepcontext: Context-encoding neural pathways for 3d holistic scene understanding. *arXiv preprint arXiv:1603.04922*, 2016.

[17] H. S. Koppula, A. Anand, T. Joachims, and A. Saxena. Semantic labeling of 3d point clouds for indoor scenes. In *Advances in Neural Information Processing Systems*, pages 244–252, 2011.

[18] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *ICRA*, pages 1817–1824. IEEE, 2011.

[19] D. Lin, S. Fidler, and R. Urtasun. Holistic scene understanding for 3d object detection with rgbd cameras. In *ICCV*, pages 1417–1424, 2013.

[20] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, pages 740–755. Springer, 2014.

[21] T. Malisiewicz, A. Gupta, and A. A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *2011 ICCV*, pages 89–96. IEEE, 2011.

[22] A. F. Martins, M. A. Figeuiredo, P. M. Aguiar, N. A. Smith, and E. P. Xing. An augmented lagrangian approach to constrained map inference. 2011.

[23] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, pages 779–788, 2016.

[24] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[25] X. Ren, L. Bo, and D. Fox. Rgb-(d) scene labeling: Features and algorithms. In *CVPR*, pages 2759–2766. IEEE, 2012.

[26] Z. Ren and E. B. Sudderth. Three-dimensional object detection and layout prediction using clouds of oriented gradients. CVPR, 2016.

[27] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.

[28] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, pages 746–760. Springer, 2012.

[29] S. Song, S. P. Lichtenberg, and J. Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *CVPR*, pages 567–576, 2015.

[30] S. Song and J. Xiao. Sliding shapes for 3d object detection in depth images. In *ECCV*, pages 634–651. Springer, 2014.

[31] S. Song and J. Xiao. Deep sliding shapes for amodal 3d object detection in rgb-d images. *arXiv preprint arXiv:1511.02300*, 2015.

[32] S. Tang, X. Wang, X. Lv, T. X. Han, J. Keller, Z. He, M. Skubic, and S. Lao. Histogram of oriented normal vectors for object recognition with a depth sensor. In *ACCV*, pages 525–538. Springer, 2012.

[33] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *IJCV*, 104(2):154–171, 2013.

[34] M. J. Wainwright, M. I. Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.