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.

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
greatly increases the search space in 3D, and thus slows
the search process down. (ii) Another reason is the in-
complete sparse data available in 3D point clouds generated
by a single RGB-D image, which suffers from weak 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 chan-
nels 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 segmenta-
tions. Information that can be encoded in 3D include den-
sities, normals, gradients, signed distance functions, among
others. Nonetheless, all these 3D voxelization-based tech-
niques suffer from the large amount of missing 3D informa-
tion, whereby the observable points in a scene only consti-
tute 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 de-
sired 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 us-
ing 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 perfor-
mance 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 representa-
tive methods that address this problem, namely DPM (de-
formable parts model) [3] and Selective Search [33] before
the ubiquity of deep learning based methods, as well as, re-
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 tar-
get 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 meth-
ods [18] [19] [15] [32], which study object detection in the
presence of depth information. Other techniques seman-
tically 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 im-
ages 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 de-
tectors 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 ac-
cording to hand-crafted features. The method proposed in
[11] places 3D bounding boxes around objects in the con-
text 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 pro-
pose candidate boxes and then classifies them by forming
a conditional random field (CRF) that integrates informa-
tion 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 classifica-
tion 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 CNNs 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 \cite{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 ~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 \cite{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 \cite{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 \cite{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) \cite{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 \cite{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

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

where $\mathbf{N}$ is the matrix containing the normals at every 3D point, $\lambda$ 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
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 \( l_i \in \{0, 1, ..., n_l\} \).
The above problem can be transformed into a linear program (LP) by introducing the local marginal variables \( p_u \), \( p_b \), and \( p \) associated with unary term.

\[ \arg \max \prod_{i=1}^{n_u} p_u(l_i) \prod_{(i,j) \in E} \frac{1}{p(l_i, l_j)} \]

Here \( n_i \) 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

\[ \log p_{U,B}(l) \propto \sum_{i=1}^{n_u} U_i(l_i) + \sum_{(i,j) \in E} B(l_i, l_j) \]

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

\[ L = \arg \max \prod_{i=1}^{n_u} p_u(l_i) \prod_{(i,j) \in E} \frac{1}{p(l_i, l_j)} \]

The above problem can be transformed into a linear program (LP) by introducing the local marginal variables \( p_u \) and \( p_b \). We then 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 curves 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

\[ \alpha = \frac{d - o}{d - o} \]
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.
### 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 7x faster than DSS and is slightly better in terms of accuracy. The final version is 4.7x 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.

### 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


