# ImVoxelNet: Image to Voxels Projection for Monocular and Multi-View General-Purpose 3D Object Detection

Danila Rukhovich[1,2], Anna Vorontsova[1], Anton Konushin[1,2]
[1]Samsung AI Center Moscow; [2]Lomonosov Moscow State University
{d.rukhovich, a.vorontsova, a.konushin}@samsung.com

## Abstract

*In this paper, we introduce the task of multi-view RGB-based 3D object detection as an end-to-end optimization problem. To address this problem, we propose ImVoxelNet, a novel fully convolutional method of 3D object detection based on posed monocular or multi-view RGB images. The number of monocular images in each multi-view input can variate during training and inference; actually, this number might be unique for each multi-view input. ImVoxelNet successfully handles both indoor and outdoor scenes, which makes it general-purpose. Specifically, it achieves state-of-the-art results in car detection on KITTI (monocular) and nuScenes (multi-view) benchmarks among all methods that accept RGB images. Moreover, it surpasses existing RGB-based 3D object detection methods on the SUN RGB-D dataset. On ScanNet, ImVoxelNet sets a new benchmark for multi-view 3D object detection. The source code and the trained models are available at* `https://github.com/saic-vul/imvoxelnet`.

## 1. Introduction

RGB images are an affordable and universal data source; therefore, RGB-based 3D object detection has been actively investigated in recent years. RGB images provide visual clues about the scene and its objects, yet they do not contain explicit information about the scene geometry and the absolute scale of the data. By virtue of that, detecting 3D objects from the RGB images is an ill-posed task. Given a monocular image, deep learning-based 3D object detection methods can only deduce the scale of the data. Moreover, the scene geometry cannot be unambiguously derived from the RGB images since some areas may be invisible. However, using several posed images might help obtain more information about the scene than a monocular RGB image. Accordingly, some 3D object detection methods [34, 32] run multi-view inference. These methods obtain predictions on each monocular RGB image independently, then aggregate these predictions.

In contrast, we use multi-view inputs not only for inference but also for training. During both training and inference, the proposed method accepts posed multi-view inputs with an arbitrary number of views; this number might be unique for each multi-view input. Besides, our method can accept posed monocular inputs (treated as a special case of multi-view inputs). Furthermore, it works surprisingly well on monocular benchmarks.

All RGB-based 3D object detection methods are designed to be indoor or outdoor and work under certain assumptions about the scene and the objects. For instance, outdoor methods are typically evaluated on cars. In general, cars are of similar size, they are located on the ground, and their projections onto the Bird's Eye View (BEV) do not intersect. Accordingly, a BEV-plane projection contains much information on the 3D location of a car. So, a common approach in outdoor 3D object detection is to reduce a 3D object detection in a point cloud to a 2D object detection in the BEV plane. At the same time, indoor objects might have different heights and be randomly located in space, so their projections onto the floor plane provide little information about their 3D positions. Overall, the design of RGB-based 3D object detection methods tends to be domain-specific.

To accumulate information from multiple inputs, we construct a voxel representation of the 3D space. We use this unified approach to detect objects in both indoor and outdoor scenes: we only choose between an indoor and outdoor head, while the meta-architecture remains the same.

In the proposed method, final predictions are obtained from 3D feature maps, which corresponds to the formulation of the point cloud-based detection problem. On this basis, we use off-the-shelf necks and heads from point cloud-based object detectors with no modifications.

Our contribution is three-fold:

- As far as we know, we are the first to formulate a task of end-to-end training for multi-view 3D object detection based on posed RGB images only.

- We propose a novel fully convolutional 3D object detector that works in both monocular and multi-view settings.

- With domain-specific heads, the proposed method achieves state-of-the-art results for both indoor and outdoor datasets.

## 2. Related Works

### 2.1. Multi-view Scene Understanding

Many scene understanding methods accept multi-view inputs. For instance, some scene understanding sub-tasks can only be solved given multi-view inputs. For example, the SLAM task implies reconstructing 3D scene geometry and estimating camera poses given a sequence of frames. Structure-from-Motion (SfM) approaches are designed to estimate camera poses and intrinsics from an unordered set of images, whereas Multi-View Stereo (MVS) methods use SfM outputs to build a 3D point cloud.

Other scene understanding sub-tasks might be reformulated to be multi-view. Several methods that use multi-view inputs to address these tasks have been proposed recently. For instance, 3D-SIS [13] performs 3D instance segmentation based on a set of RGB-D inputs. MVPointNet [17] uses multi-view RGB-D inputs for 3D semantic segmentation. Atlas [26] processes several monocular RGB images to perform 3D semantic segmentation and TSDF reconstruction jointly.

### 2.2. 3D Object Detection.

**Point cloud-based.** Point clouds are three-dimensional, so it seems natural to employ a 3D convolutional network for detection. However, this approach requires exhaustive computation that causes slow inference on large outdoor scenes. Recent outdoor methods [38, 19] decrease the runtime by projecting the 3D point cloud to the BEV plane. The common practice in point cloud processing is to subdivide a point cloud into voxels. The projection onto the BEV plane implies that all voxels in each vertical column should be encoded into a fixed-length feature map. Then, this pseudo-image can be passed to a 2D object detection network to obtain final predictions.

Indoor object detection methods generate object proposals for each point in a point cloud. However, some indoor objects are not convex, so the geometrical center of an indoor object may not belong to this object (e.g., the center of a table or a chair might be in between legs). Accordingly, an object proposal given by a single center point might be irrelevant, so indoor methods use deep Hough voting to generate proposals [28, 29, 40].

**Stereo-based.** Despite accepting more than one image, stereo-based methods cannot be considered multi-view as they use two images. In contrast, multi-view methods can process an arbitrary amount of inputs. Moreover, camera poses might be arbitrary for multi-view inputs, and for stereo inputs, the relative transformation between two cameras is known precisely and remains fixed while recording. This makes it possible to perform stereo reconstruction by estimating optical flow between the left and right images. Stereo-based methods rely heavily on the stereo assumptions, e. g., 3DOP [6] uses stereo reconstruction to generate object proposals, while TLNet [31] runs triangulation to merge proposals obtained for left and right images independently. Stereo R-CNN [21] generates object proposals given both left and right images, then estimates object location by triangulating keypoints.

**Monocular-based.** Mono3D [7] generates 3D anchors by aggregating clues from semantic maps, visible contours of the objects, and location priors via a complex energy function. Deep3DBox [25] uses discretization to estimate the orientation of each object and derives its 3D pose from constraints between 2D and 3D bounding boxes. MonoGRNet [30] decomposes the 3D object detection problem into sub-tasks, namely object distance estimation, object location estimation, and object corners estimation. These subtasks are solved by separate networks, trained first stagewise then altogether to refine 3D bounding boxes.

Other methods, e.g., [4, 14, 27], exploit 2D detection and lift information from 2D to 3D. [15, 14, 27] extend 2D detection network with a 3D branch that regresses object pose. Some methods make use of external data sources, e.g., DeepMANTA [4] uses an iterative coarse-to-fine algorithm of generating 2D object proposals, which are used to select a CAD model. 3D-RCNN [18] also performs 2D detection and matches the outputs to 3D models. Then, it uses a render-and-compare approach to recover the shape and pose of an object.

Monocular indoor 3D object detection is a less explored problem, with only SUN RGB-D [36] benchmark existing. This benchmark implies that indoor 3D object detection is a sub-task of total scene understanding. Beside detecting 3D objects, [15, 14, 27] estimate camera poses and room layouts. The most recent Total3DUnderstanding [27] reconstructs object meshes using an attention mechanism to consider relationships between objects.

Some outdoor 3D object detection methods [34, 32] are evaluated on the nuScenes [3] dataset on multi-view inputs. Specifically, these methods infer on each monocular RGB image, then aggregate the outputs. Aggregation is an inevitable part of the pipeline; however, doing this on the latest stage is controversial, as spatial information might not be exploited as effectively as possible.

So, none of the existing methods formulate 3D object detection given multiple RGB images as an end-to-end optimization problem.

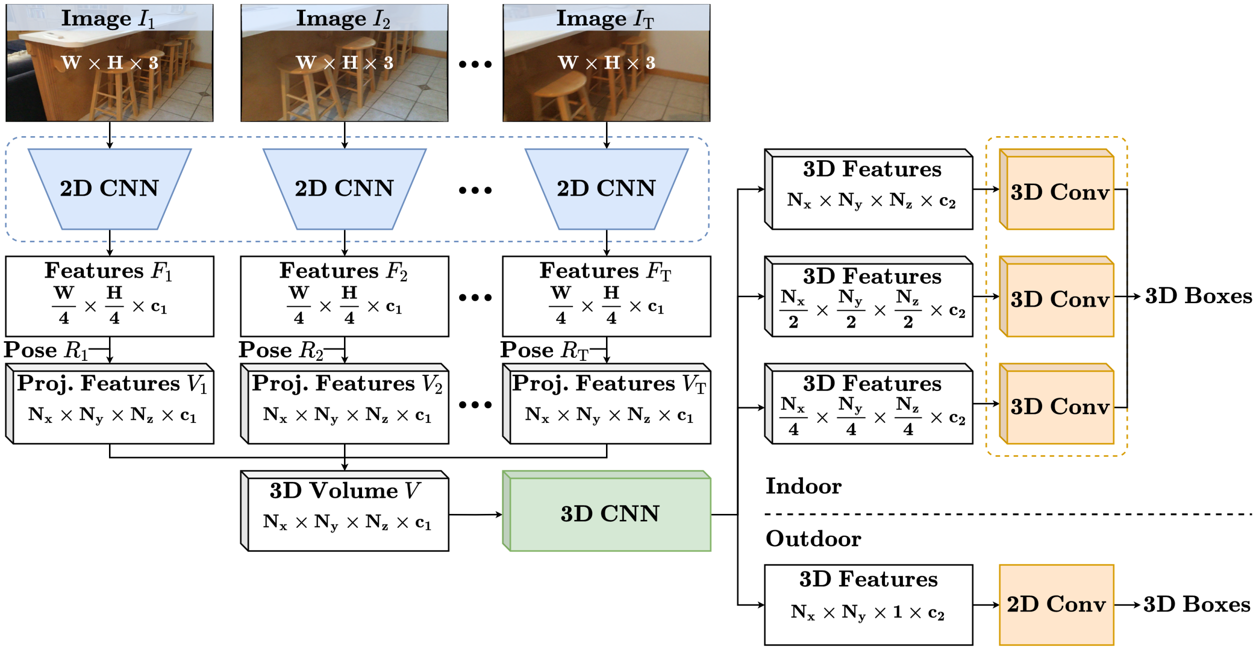
Image $I_1$
W × H × 3
Image $I_2$
W × H × 3
Image $I_T$
W × H × 3
2D CNN
2D CNN
2D CNN
Features $F_1$
$\frac{W}{4} \times \frac{H}{4} \times c_1$
Features $F_2$
$\frac{W}{4} \times \frac{H}{4} \times c_1$
Features $F_T$
$\frac{W}{4} \times \frac{H}{4} \times c_1$
Pose $R_1$
Pose $R_2$
Pose $R_T$
Proj. Features $V_1$
$N_x \times N_y \times N_z \times c_1$
Proj. Features $V_2$
$N_x \times N_y \times N_z \times c_1$
Proj. Features $V_T$
$N_x \times N_y \times N_z \times c_1$
3D Volume $V$
$N_x \times N_y \times N_z \times c_1$
3D CNN
3D Features
$N_x \times N_y \times N_z \times c_2$
3D Features
$\frac{N_x}{2} \times \frac{N_y}{2} \times \frac{N_z}{2} \times c_2$
3D Features
$\frac{N_x}{4} \times \frac{N_y}{4} \times \frac{N_z}{4} \times c_2$
3D Conv
3D Conv
3D Conv
3D Boxes
Indoor
Outdoor
3D Features
$N_x \times N_y \times 1 \times c_2$
2D Conv
3D Boxes

Figure 1. The general scheme of the proposed ImVoxelNet. Dashed lines around network blocks denote that network weights are shared across multiple inputs.

## 3. Proposed Method

Our method accepts an arbitrary-sized set of RGB inputs along with camera poses. First, we extract features from the given images using a 2D convolutional backbone. Then, we project the obtained image features to a 3D voxel volume. For each voxel, the projected features from several images are aggregated via a simple element-wise averaging. Next, the voxel volume with assigned features is passed to a 3D convolutional network referred to as *neck*. The outputs of the neck serve as inputs to the last few convolutional layers (*head*) that predict bounding box features for each anchor. The resulting bounding boxes are parameterized as $(x, y, z, w, h, l, \theta)$, where $(x, y, z)$ are the coordinates of the center, $w, h, l$ are for width, height, and length, and $\theta$ is the rotation angle around $z$-axis. The general scheme of the proposed method is depicted in Fig. 1.

2D features projection and 3D neck network have been proposed in [26, 13]. First, we briefly outline these steps. Then, we introduce a novel multi-scale 3D head designed for indoor detection.

### 3.1. 3D Volume Construction

Let $I_t \in \mathbb{R}^{W \times H \times 3}$ be the $t$-th image in a set of $T$ images. Here, $T > 1$ in case of multi-view inputs and $T = 1$ for single-view inputs. Following [26], we first extract 2D features from passed inputs using a pretrained 2D backbone. It outputs four feature maps of shapes $\frac{W}{4} \times \frac{H}{4} \times c_0$, $\frac{W}{8} \times \frac{H}{8} \times 2c_0$, $\frac{W}{16} \times \frac{H}{16} \times 4c_0$, and $\frac{W}{32} \times \frac{H}{32} \times 8c_0$. We aggregate the obtained feature maps via Feature Pyramid Network (FPN), which outputs one tensor $F_t$ of shape $\frac{W}{4} \times \frac{H}{4} \times c_1$. $c_0$ and $c_1$ are backbone-specific; actual values are present in 4.2.

For t-th input, the extracted 2D features $F_t$ are then projected into a 3D voxel volume $V_t \in \mathbb{R}^{N_x \times N_y \times N_z \times c_1}$. We set the $z$-axis to be perpendicular to the floor plane, with the $x$-axis pointing forward and the $y$-axis being orthogonal to both $x$ and $z$-axes. For each dataset, there are known spatial limits for all three axes, estimated empirically in [40, 19, 26]. Let us denote these limits as $x_{\text{min}}, x_{\text{max}}, y_{\text{min}}, y_{\text{max}}, z_{\text{min}}, z_{\text{max}}$. For a fixed voxel size $s$, spatial constraints can be formulated as $N_x s = x_{\text{max}} - x_{\text{min}}$, $N_y s = y_{\text{max}} - y_{\text{min}}$, and $N_z s = z_{\text{max}} - z_{\text{min}}$. We use a pinhole camera model, which determines the correspondence between 2D coordinates $(u, v)$ in feature map $F_t$ and 3D coordinates $(x, y, z)$ in volume $V_t$:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \Pi \begin{bmatrix} \frac{1}{4} & 0 & 0 \\ 0 & \frac{1}{4} & 0 \\ 0 & 0 & 1 \end{bmatrix} K R_t \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix},$$

where $K$ and $R_t$ are the intrinsic and extrinsic matrices, and $\Pi$ is a perspective mapping. After projecting 2D features, all voxels along a camera ray get filled with the same features. We also define a binary mask $M_t$ of the same shape as $V_t$, which indicates whether each voxel is inside the camera frustum. Thus, for each image $I_t$, the mask $M_t$ is defined as:

$$M_t(x,y,z) = \begin{cases} 1, & \text{if } 0 \leq u < \frac{W}{4} \text{ and } 0 \leq v < \frac{H}{4} \\ 0, & \text{otherwise.} \end{cases}$$

Then, we project $F_t$ for each valid voxel in a volume $V_t$:

$$V_t(x,y,z) = \begin{cases} F_t(u,v), & \text{if } M_t(x,y,z) = 1 \\ 0, & \text{otherwise.} \end{cases}$$

The aggregated binary mask $M$ is a sum of $M_1, \ldots, M_t$:

$$M(x,y,z) = \begin{cases} \sum_t M_t(x,y,z), & \text{if } \sum_t M_t(x,y,z) > 0 \\ 1, & \text{otherwise.} \end{cases}$$

Finally, we obtain the 3D volume $V$ by averaging projected features in volumes $V_1, \ldots, V_t$ across valid voxels:

$$V = \frac{1}{M} \sum_t M_t V_t.$$

## 3.2. 3D Feature Extraction

**Indoor.** Following [26, 13], we pass the voxel volume $V$ through a 3D convolutional encoder-decoder network to refine the features. For indoor scenes, we use an encoder-decoder architecture from [26]. However, with over 48 3D convolutional layers, the original network is computationally heavy and slow on inference. For a better performance, we simplify the network by reducing the number of time-consuming 3D convolutional layers. The simplified encoder has only three downsampling residual blocks, each with three 3D convolutional layers. The simplified decoder consists of three upsampling blocks, and each upsampling block is made up with a transposed 3D convolutional layer with stride 2 followed by another 3D convolutional layer. The decoder branch outputs three feature maps of the following shapes: $\frac{N_x}{4} \times \frac{N_y}{4} \times \frac{N_z}{4} \times c_2$, $\frac{N_x}{2} \times \frac{N_y}{2} \times \frac{N_z}{2} \times c_2$, and $N_x \times N_y \times N_z \times c_2$. For the actual value of $c_2$, see 4.2.

**Outdoor.** Outdoor methods [35, 19, 38] reduce 3D object detection in 3D space to 2D object detection in the BEV plane. In these methods, both the neck and head are composed of 2D convolutions. The outdoor head accepts a 2D feature map, so we should obtain a 2D representation of a constructed 3D voxel volume to use in our method. In order to do that, we use the encoder part of the encoder-decoder architecture from [26]. After passing through several 3D convolutional and downsampling layers of this encoder, a voxel volume $V$ of shape $N_x \times N_y \times N_z \times c_1$ is mapped to the tensor of shape $N_x \times N_y \times c_2$.

## 3.3. Detection Heads

ImVoxelNet constructs a 3D voxel representation of the space; thus, it can use the head from point cloud-based 3D object detection methods. Therefore, instead of time-consuming custom architecture implementation, one can employ state-of-the-art methods with no modifications. However, the design of heads significantly differs for outdoor [19, 38] and indoor [28, 29] methods.

### 3.3.1 Outdoor Head

We reformulate outdoor 3D object detection as 2D object detection in the BEV plane following the common practice. We use the 2D anchor head that appeared to be efficient [19, 38] on KITTI [11] and nuScenes [3] datasets. Since outdoor 3D detection methods are evaluated on cars, all objects are of a similar scale and belong to the same category. For single-scale and single-class detection, the head consists of two parallel 2D convolutional layers. One layer estimates class probability, while the other regresses seven parameters of the bounding box.

**Input.** The input is a tensor of shape $N_x \times N_y \times c_2$.

**Output.** For each 2D BEV anchor, the head returns a class probability $p$ and a 3D bounding box as a 7-tuple:

$$\Delta x = \frac{x^{\mathsf{gt}} - x^{\mathsf{a}}}{d^{\mathsf{a}}}, \Delta y = \frac{y^{\mathsf{gt}} - y^{\mathsf{a}}}{d^{\mathsf{a}}}, \Delta z = \frac{z^{\mathsf{gt}} - z^{\mathsf{a}}}{d^{\mathsf{a}}},$$

$$\Delta w = \log \frac{w^{\mathsf{gt}}}{w^{\mathsf{a}}}, \Delta l = \log \frac{l^{\mathsf{gt}}}{l^{\mathsf{a}}}, \Delta h = \log \frac{h^{\mathsf{gt}}}{h^{\mathsf{a}}},$$

$$\Delta \theta = \sin(\theta^{\mathsf{gt}} - \theta^{\mathsf{a}}).$$

Here $\cdot^{\mathsf{gt}}$ and $\cdot^{\mathsf{a}}$ are the ground truth and anchor boxes, respectively. The length of the bounding box diagonal $d^{\mathsf{a}} = \sqrt{(w^{\mathsf{a}})^2 + (l^{\mathsf{a}})^2}$. $z_{\mathsf{a}}$ is constant for all anchors since they are located in the BEV plane.

**Loss.** We use the loss function introduced in SECOND [38]. The total outdoor loss consists of several loss terms, namely smooth mean absolute error as a location loss $L_{\mathsf{loc}}$, focal loss for classification $L_{\mathsf{cls}}$, and cross-entropy loss for direction $L_{\mathsf{dir}}$. Overall, we can formulate the outdoor loss as

$$L_{\mathsf{outdoor}} = \frac{1}{n_{\mathsf{pos}}} (\lambda_{\mathsf{loc}} L_{\mathsf{loc}} + \lambda_{\mathsf{cls}} L_{\mathsf{cls}} + \lambda_{\mathsf{dir}} L_{\mathsf{dir}}),$$

where $n_{\mathsf{pos}}$ is the number of positive anchors, $\lambda_{\mathsf{loc}} = 2$, $\lambda_{\mathsf{cls}} = 1$, $\lambda_{\mathsf{dir}} = 0.2$.

### 3.3.2 Indoor Head

All modern indoor 3D object detection methods [28, 29, 40] perform deep Hough voting for sparse point cloud representation. In contrast, we follow [26, 13] and use dense voxel representation of intermediate features. To the best of our knowledge, there is no dense 3D multi-scale head for 3D object detection. We construct such a head inspired by a 2D detection method FCOS [37]. An original FCOS head accepts 2D features from FPN and estimates 2D bounding boxes via 2D convolutional layers. To adapt FCOS for 3D detection, we replace 2D convolutions with 3D convolutions to process 3D inputs. Following FCOS and ATSS [39], we apply center sampling to select candidate object locations. In these works, 9 $(3 \times 3)$ candidates were chosen; since we operate in 3D space, we set a limit of 27 candidate locations per object $(3 \times 3 \times 3)$. The resulting head consists of three 3D convolutional layers for classification, location, and centerness, respectively, with weights shared across all object scales.

**Input.** A multi-scale input is composed of three tensors of shapes $\frac{N_x}{4} \times \frac{N_y}{4} \times \frac{N_z}{4} \times c_2$, $\frac{N_x}{2} \times \frac{N_y}{2} \times \frac{N_z}{2} \times c_2$, and $N_x \times N_y \times N_z \times c_2$.

**Output.** For each 3D location $(x^{\mathsf{a}}, y^{\mathsf{a}}, z^{\mathsf{a}})$ and each of three scales, the head estimates a class probability $p$, a centerness $c$, and a 3D bounding box as a 7-tuple:

$$\Delta x_{\mathsf{min}} = x^{\mathsf{gt}}_{\mathsf{min}} - x^{\mathsf{a}}, \Delta x_{\mathsf{max}} = x^{\mathsf{gt}}_{\mathsf{max}} - x^{\mathsf{a}},$$

$$\Delta y_{\mathsf{min}} = y^{\mathsf{gt}}_{\mathsf{min}} - y^{\mathsf{a}}, \Delta y_{\mathsf{max}} = y^{\mathsf{gt}}_{\mathsf{max}} - y^{\mathsf{a}},$$

$$\Delta z_{\mathsf{min}} = z^{\mathsf{gt}}_{\mathsf{min}} - z^{\mathsf{a}}, \Delta z_{\mathsf{max}} = z^{\mathsf{gt}}_{\mathsf{max}} - z^{\mathsf{a}}, \theta.$$

Here, $x^{\mathsf{gt}}_{\mathsf{min}}, x^{\mathsf{gt}}_{\mathsf{max}}, y^{\mathsf{gt}}_{\mathsf{min}}, y^{\mathsf{gt}}_{\mathsf{max}}, z^{\mathsf{gt}}_{\mathsf{min}}, z^{\mathsf{gt}}_{\mathsf{max}}$ denote the minimum and maximum coordinates along axes of a ground truth bounding box.

**Loss.** We adapt the loss function used in the original FCOS [37]. It consists of focal loss for classification $L_{\mathsf{cls}}$, cross-entropy loss for centerness $L_{\mathsf{cntr}}$, and IoU loss for location $L_{\mathsf{loc}}$. Since we address the 3D detection task instead of the 2D detection task, we replace 2D IoU loss with rotated 3D IoU loss [41]. In addition, we update ground truth centerness with the third dimension. The resulting indoor loss can be written as

$$L_{\mathsf{indoor}} = \frac{1}{n_{\mathsf{pos}}}(L_{\mathsf{loc}} + L_{\mathsf{cls}} + L_{\mathsf{cntr}}),$$

where $n_{\mathsf{pos}}$ is the number of positive 3D locations.

### 3.4. Extra 2D Head

In some indoor benchmarks, the 3D object detection task is formulated as a sub-task of scene understanding. Accordingly, evaluation protocols imply solving various scene understanding tasks rather than only estimating 3D bounding boxes. Following [15, 14, 27], we predict camera rotations and room layouts. Similar to [27], we add a simple head for joint $R_t$ and 3D layout estimation. This extra head consists of two parallel branches: two fully connected layers output room layout and the other two fully connected layers estimate camera rotation.

**Input.** The input is a single tensor of shape $8c_0$, obtained through global average pooling of the backbone output.

**Output.** The head outputs camera pose as a tuple of pitch $\beta$ and roll $\gamma$ and a 3D layout box as a 7-tuple $(x, y, z, w, l, h, \theta)$. As [27], we set yaw angle and shift to zeros.

**Loss.** We modify losses used in [27] to make them consistent with the losses used to train a detection head. Accordingly, we define layout loss $L_{\mathsf{layout}}$ as rotated 3D IoU loss between predicted and ground truth layout boxes; this is the same loss as we use in 3.3.2. For camera rotation estimation, we use $L_{\mathsf{pose}} = |\sin(\beta^{\mathsf{gt}} - \beta)| + |\sin(\gamma^{\mathsf{gt}} - \gamma)|$ similar to 3.3.1. Overall, the extra loss can be formulated as

$$L_{\mathsf{extra}} = \lambda_{\mathsf{layout}} L_{\mathsf{layout}} + \lambda_{\mathsf{pose}} L_{\mathsf{pose}},$$

where $\lambda_{\mathsf{layout}} = 0.1$ and $\lambda_{\mathsf{pose}} = 1.0$.

## 4. Experiments

### 4.1. Datasets

We evaluate the proposed method on four datasets: indoor ScanNet [9] and SUN RGB-D [36], and outdoor KITTI [11] and nuScenes [3]. SUN RGB-D and KITTI are benchmarked in monocular mode, while for ScanNet and nuScenes, we address the detection problem in multi-view formulation.

**KITTI.** The KITTI object detection dataset [11] is the most decisive outdoor benchmark for monocular 3D object detection. It consists of 3711 training, 3768 validation and 7518 test images. The common practice [34, 23] is to report results on validation subset and submit test predictions to an open leaderboard. All 3D object annotations have a difficulty level: *easy*, *moderate*, and *hard*. A 3D object detection method is assessed according to the results on *moderate* objects from the test set. Following [34, 23], we evaluate our method only on objects of the *car* category.

**nuScenes.** The nuScenes dataset [3] provides data for developing algorithms addressing self-driving-related tasks. It contains LiDAR point clouds, RGB images captured by six cameras, accompanied by IMU and GPS measurements. The dataset covers 1000 video sequences, each recorded for 20 seconds, totalling 1.4 million images and 390 000 point clouds. Training split covers 28 130 scenes, and validation split contains 6019 scenes. The annotation contains 1.4 million objects divided into 23 categories. Following [34], the accuracy of 3D detection is measured only on *car* category. In this benchmark, not only the average precision (AP) metric but average translation error (ATE), average scale error (ASE), and average orientation error (AOE) are calculated as well.

**SUN RGB-D.** SUN RGB-D [36] is one of the first and most well-known indoor 3D datasets. It contains 10 335 images captured in various indoor places alongside corresponding depth maps obtained with four different sensors and camera poses. The training split is composed of 5285 frames, while the rest 5050 frames comprise the validation

| Dataset | $x_{\text{min}}$ | $x_{\text{max}}$ | $y_{\text{min}}$ | $y_{\text{max}}$ | $z_{\text{min}}$ | $z_{\text{max}}$ | $s$ |
|---|---|---|---|---|---|---|---|
| KITTI | -39.68 | 39.68 | 0 | 69.12 | -2.92 | 0.92 | 0.32 |
| nuScenes | -49.92 | 49.92 | -49.92 | 49.92 | -2.92 | 0.92 | |
| SUN RGB-D | -3.2 | 3.2 | 0 | 6.4 | -2.28 | 0.28 | 0.16 |
| ScanNet | -3.2 | 3.2 | -3.2 | 3.2 | -1.28 | 1.28 | |

Table 1. Implementation details. Axis limits and voxel size $s$ are measured in meters.

subset. The annotation includes 58 657 objects. For each frame, a room layout is provided.

**ScanNet.** The ScanNet dataset [9] contains 1513 scans covering over 700 unique indoor scenes, out of which 1201 scans belong to a training split, and 312 scans are used for validation. Overall, this dataset contains over 2.5 million images with corresponding depth maps and camera poses, alongside reconstructed point clouds with 3D semantic annotation. We estimate 3D bounding boxes from semantic point clouds following the standard protocol [28]. The resulting object bounding boxes are axis-aligned, so we do not predict the rotation angle $\theta$ for ScanNet.

## 4.2. Implementation Details

**3D Volume.** We use ResNet-50 [12] as a feature extractor. Accordingly, the number of convolutions in the first convolutional block $c_0$ equals 256. We set both the 3D volume feature size $c_1$ and the ouput feature size $c_2$ to 256 as proposed in [19, 38].

Indoor and outdoor scenes are of different absolute scales. Therefore, we choose the spatial sizes of the feature volume for each dataset considering the data domain. We use the values provided in previous works [26, 19, 38, 35], as shown in Tab. 1. Thus, using anchor settings of the 3D head in [19, 35], we set voxel size $s$ as 0.32 meters for outdoor datasets. Minimal and maximal values for all three axes for outdoor datasets also follow the point cloud ranges for *car* class in [19, 35]. For selecting indoor dataset constraints we follow [26], where the room size is $6.4 \times 6.4 \times 2.56$ meters. The only change is that we are increasing voxels size $s$ from $0.04$ to $0.16$ to increase memory efficiency.

**Training.** During training, we optimize $L_{\text{indoor}}$ for indoor datasets and $L_{\text{outdoor}}$ for outdoor datasets, unless told otherwise. We use Adam optimizer with an initial learning rate set to $0.0001$ and weight decay of $0.0001$. The implementation is based on the MMDetection framework [5] and uses its default training settings. The network is trained for 12 epochs, and the learning rate is reduced by ten times after the 8th and 11th epoch. For ScanNet, SUN RGB-D, and KITTI, the network sees each scene three times every training epoch. We use 8 Nvidia Tesla P40 GPUs for training, distributing one scene (multi-view scenario) or four images (monocular scenario) per GPU. We randomly apply horizontal flip and resize inputs in monocular experiments by no more than 25% of their original resolution. Moreover, in indoor scenes, we can augment 3D voxel representations similar to point cloud-based methods, so we randomly shift a voxel grid center by at most 1m along each axis.

**Inference.** During inference, outputs are filtered with a Rotated NMS algorithm, which is applied to objects projections onto the ground plane.

## 4.3. Results

First, we report the results of detecting cars on outdoor KITTI and nuScenes benchmarks. Then, we discuss the results of multi-class 3D object detection on SUN RGB-D and ScanNet indoor datasets.

**KITTI.** We present the results of monocular *car* detection on KITTI in Tab. 2. ImVoxelNet achieves the best *moderate* AP on the *test* split, which is the main metric in the KITTI benchmark. Moreover, our method surpasses previous state-of-the-art by 6% $\text{AP}_{\text{3D}}$ and 4% $\text{AP}_{\text{BEV}}$ for *easy* objects. Overall, ImVoxelNet is superior in terms of almost all metrics on both *test* and *val* splits.

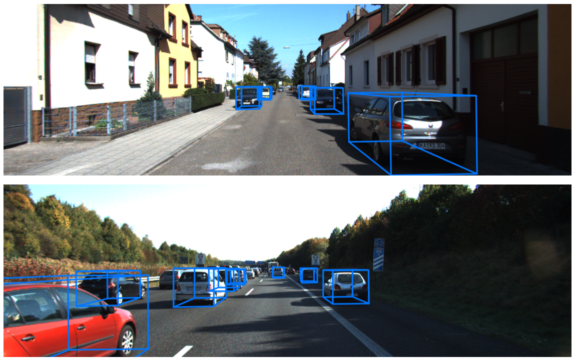

Figure 2. Visualization of object detection results for monocular images from validation subset of the KITTI dataset.

**nuScenes.** For nuScenes, unlike other methods that only run inference on images from 6 onboard cameras, ImVoxelNet uses multi-view inputs for training. As shown in Tab. 3, the proposed method outperforms MonoDIS [34] by more than 1% of mean AP, which is the main metric. According to AP@0.5, ImVoxelNet outputs almost twice as many highly accurate estimates comparing to MonoDIS. For car detection, two boxes might have IoU = 0 when a center distance exceeds 1 meter. By that, AP@1.0m, AP@2.0m, and AP@4.0m might be calculated for non-intersecting bounding boxes, which seems counter-intuitive (e.g., for the KITTI dataset, only boxes with IoU >0.7 are considered to be true positive). Hence, we argue that AP@0.5 is the most decisive metric.

Moreover, we report values of ATE, ASE, and AOE metrics. As represented in the Tab. 3, ImVoxelNet has at least 0.09 meters smaller ATE than other monocular methods.

**SUN RGB-D.** We compare ImVoxelNet with existing

| Method | Depth | $AP_{3D}$@0.7 (*val*/*test*) | | | $AP_{BEV}$@0.7 (*val*/*test*) | | |
|---|---|---|---|---|---|---|---|
| | | Easy | Moderate | Hard | Easy | Moderate | Hard |
| MonoFENet[1] | ✓ | 17.54 / 8.34 | 11.16 / 5.14 | 9.74 / 4.10 | 30.21 / 17.03 | 20.47 / 11.03 | 17.58 / 9.05 |
| AM3D[24] | ✓ | 32.23 / 16.50 | 21.09 / 10.74 | 17.26 / 9.52 | 43.75 / 25.03 | 28.39 / 17.32 | 23.87 / 14.91 |
| D4LCN[10] | ✓ | 26.97 / 16.65 | 21.71 / 11.72 | 18.22 / 9.51 | 34.82 / 22.51 | 25.83 / 16.02 | 23.53 / 12.55 |
| OFTNet[32] | ✗ | 4.47 / 1.32 | 3.27 / 1.61 | 3.29 / 1.00 | 11.06 / 7.16 | 8.79 / 5.69 | 8.91 / 4.61 |
| GS3D[20] | ✗ | 13.46 / 4.47 | 10.97 / 2.90 | 10.38 / 2.47 | – / 8.41 | – / 6.08 | – / 4.94 |
| MonoGRNet[30] | ✗ | 13.88 / 9.61 | 10.19 / 5.74 | 7.62 / 4.25 | – / 18.19 | – / 11.17 | – / 8.73 |
| MonoDIS[34] | ✗ | 18.05 / 10.37 | 14.98 / 7.94 | 13.42 / 6.40 | 24.26 / 17.23 | 18.43 / 13.19 | 16.95 / 11.12 |
| SMOKE[23] | ✗ | 14.76 / 14.03 | 12.85 / 9.76 | 11.50 / 7.84 | 19.99 / 20.83 | 15.61 / 14.49 | 15.28 / 12.75 |
| M3D-RPN[2] | ✗ | 20.27 / 14.76 | 17.06 / 9.71 | 15.21 / 7.42 | 25.94 / 21.02 | 21.18 / 13.67 | 17.90 / 10.23 |
| RTM3D[22] | ✗ | 20.77 / 14.41 | 16.86 / 10.34 | **16.63** / 8.77 | 25.56 / 19.17 | 22.12 / 14.20 | **20.91** / 11.99 |
| ImVoxelNet | ✗ | **24.54 / 17.15** | **17.80 / 10.97** | 15.67 / **9.15** | **31.67 / 25.19** | **23.68 / 16.37** | 19.73 / **13.58** |

Table 2. Scores for *car* category on the KITTI dataset. The *depth* column indicates whether this modality is used for training.

| Method | RGB | PC | AP↑[%] | | | | | TP↓ | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | 0.5m | 1.0m | 2.0m | 4.0m | mean | ATE [m] | ASE[1-IoU] | AOE[rad] |
| PointPillar[19] | ✗ | ✓ | 55.5 | 71.8 | 76.1 | 78.6 | 70.5 | 0.27 | 0.17 | 0.19 |
| OFTNet [32, 34] | ✓ | ✗ | – | – | 27.0 | – | – | 0.65 | 0.16 | 0.18 |
| MonoDIS [34] | ✓ | ✗ | 10.7 | 37.5 | **69.0** | **85.7** | 50.7 | 0.61 | **0.15** | **0.08** |
| ImVoxelNet | ✓ | ✗ | **19.3** | **44.8** | 66.3 | 77.0 | **51.8** | **0.52** | **0.15** | **0.08** |

Table 3. Scores for *car* category on the nuScenes dataset. The RGB and PC columns indicate data modalities used for both training and inference.

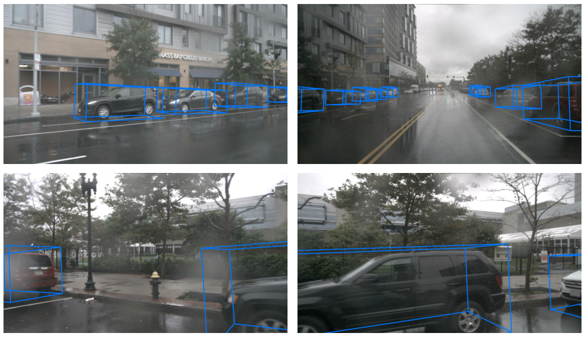

Figure 3. Visualization of object detection results for multi-view inputs from validation subset (scene *n008-2018-09-18-15-12-01-0400__15372981046*) of the nuScenes dataset.

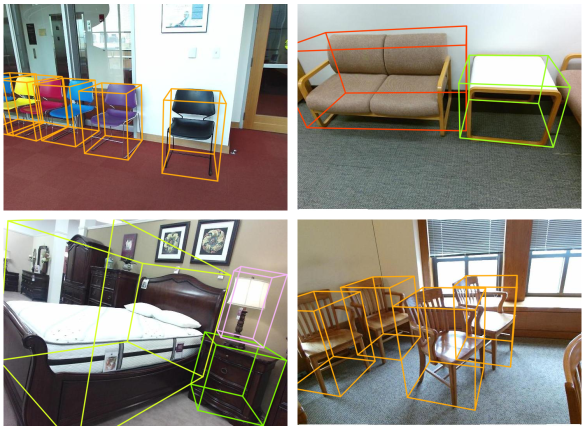

Figure 4. Visualization of object detection results for monocular images from validation subset of the SUN RGB-D dataset.

methods on the most recent monocular benchmark introduced in [27], which includes objects of NYU-37 categories [33]. Since the chosen benchmark implies estimating camera pose and layout, we optimize $L_{\text{indoor}} + L_{\text{extra}}$ for training. For a fair comparison with Total3DUnderstanding [27], we report their results without joint training since it requires the additional mesh-annotated dataset. Tab. 4 demonstrates that ImVoxelNet surpasses all previous methods by a margin exceeding 18% in terms of mAP. Furthermore, ImVoxelNet outperforms Total3DUnderstanding in both layout and camera pose estimation. We also report metrics on other benchmarks: the PerspectiveNet [16] benchmark with 30 object categories, and the VoteNet [28] benchmark with 10 categories, which is used by point cloud-based methods (see Supplementary).

**ScanNet.** We compare ImVoxelNet to existing methods on the common benchmark with 18 classes. During training, we use $T = 50$ images per scene, as was proposed in [26]. We conduct an ablation study to choose an optimal number of test images per scene (Tab. 6). We run our method five times on different samples for each number of test images and report an average result with a 0.95 confidence interval. Experiments show that the more images per test scene, the better. The most time-consuming part of the pipeline is processing a voxel volume with 3D convolutions while extracting 2D features gives a minor overhead. Consequently, with an increase in the number of test images per scene, the runtime grows sublinearly.

According to Tab. 5, ImVoxelNet still shows competitive results despite not using point clouds. Notably, it outper-

| Method | bed | chair | sofa | table | desk | dresser | nstand | sink | cabinet | lamp | mAP | Layout↑[IoU] | Pitch↓[°] | Roll↓[°] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3DGP[8] | 5.62 | 2.31 | 3.24 | 1.23 | – | – | – | – | – | – | – | 19.2 | – | – |
| HoPR[15] | 58.29 | 13.56 | 28.37 | 12.12 | 4.79 | 13.71 | 8.80 | 2.18 | 0.48 | 2.41 | 14.47 | 54.9 | 7.60 | 3.12 |
| CooP[14] | 63.58 | 17.12 | 41.22 | 26.21 | 9.55 | 4.28 | 6.34 | 5.34 | 2.63 | 1.75 | 17.80 | 56.9 | 3.28 | 2.19 |
| T3DU[27] | 59.03 | 15.98 | 43.95 | 35.28 | 23.65 | 19.20 | 6.87 | 14.40 | 11.39 | 3.46 | 23.32 | 57.6 | 3.68 | 2.59 |
| ImVoxelNet | **79.17** | **63.07** | **60.59** | **51.14** | **31.20** | **35.45** | **38.38** | **45.12** | **19.24** | **13.27** | **43.66** | **59.3** | **2.63** | **1.96** |

Table 4. AP@0.15 scores for 10 out of 37 object categories [27] from the SUN RGB-D dataset, alongside room layout and camera pose estimation metrics.

| Method | RGB | PC | cab | bed | chair | sofa | tabl | door | wind | bkshf | pic | cntr | desk | curt | fridg | showr | toil | sink | bath | ofurn | mAP |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3D-SIS[13] | ✗ | ✓ | 12.8 | 63.1 | 66.0 | 46.3 | 26.9 | 8.0 | 2.8 | 2.3 | 0.0 | 6.9 | 33.3 | 2.5 | 10.4 | 12.2 | 74.5 | 22.9 | 58.7 | 7.1 | 25.4 |
| 3D-SIS[13] | ✓ | ✓ | 19.8 | 69.7 | 66.2 | 71.8 | 36.1 | 30.6 | 10.9 | 27.3 | 0.0 | 10.0 | 46.9 | 14.1 | 53.8 | 36.0 | 87.6 | 43.0 | 84.3 | 16.2 | 40.2 |
| VoteNet[28] | ✗ | ✓ | 36.3 | 87.9 | 88.7 | 89.6 | 58.8 | 47.3 | 38.1 | 44.6 | 7.8 | 56.1 | 71.7 | 47.2 | 45.4 | 57.1 | 94.9 | 54.7 | 92.1 | 37.2 | 58.7 |
| H3DNet[40] | ✗ | ✓ | **49.4** | **88.6** | **91.8** | **90.2** | **64.9** | **61.0** | **51.9** | **54.9** | **18.6** | **62.0** | **75.9** | **57.3** | 57.2 | **75.3** | **97.9** | **67.4** | **92.5** | **53.6** | **67.2** |
| ImVoxelNet | ✓ | ✗ | 28.5 | 84.4 | 73.1 | 70.1 | 51.9 | 32.2 | 15.0 | 34.2 | 1.6 | 29.7 | 66.1 | 23.5 | **57.8** | 43.2 | 92.4 | 54.1 | 74.0 | 34.9 | 48.1 |

Table 5. AP@0.25 scores for 18 object categories from the ScanNet dataset. All methods but ImVoxelNet accept point cloud (PC) as an input.

| Images | mAP | Runtime[s] |
|---|---|---|
| 1 | 9.1 ±1.0 | 0.14 |
| 5 | 27.6 ±2.4 | 0.23 |
| 10 | 36.9 ±0.5 | 0.30 |
| 50 | 46.6 ±0.5 | 1.21 |
| 100 | **47.6** ±0.8 | 2.45 |

Table 6. mAP@0.25 scores and runtime measured in seconds per scene for different number of images per test scene from the ScanNet dataset.

forms point cloud-based 3D-SIS [13] which builds a voxel volume representation using RGB images as an additional modality.

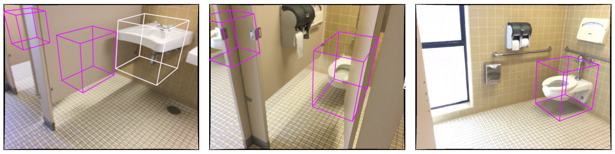

Figure 5. Visualization of object detection results for multi-view inputs from validation subset (scene *0086_00*) of the ScanNet dataset.

**Performance.** We report the inference time on the KITTI dataset in Tab. 7. All the methods were examined in the same experimental setup on a single GPU. ImVoxelNet uses computationally expensive 3D convolutions, so it is expected to be slower than the methods that rely on 2D convolutions only. In our experiments, ImVoxelNet appeared to be inferior in speed to most of the listed methods, yet the runtime differs within an order of magnitude. The listed methods use different backbones, and this affects the total speed. In ImVoxelNet, extracting features with a backbone is a simple, lightweight procedure compared to processing voxel volume with 3D convolutions. Accordingly, the choice of a backbone is negligible: experiments show that replacing ResNet-50 with a more lightweight version has a minor influence on performance.

| Method | Backbone | AP | Runtime[s] |
|---|---|---|---|
| OFTNet[32] | ResNet-18 | 3.27 | 0.50 |
| GS3D[20] | VGG-16 | 10.97 | 2.00 |
| MonoGRNet[30] | VGG-16 | 10.19 | 0.06 |
| MonoDIS[34] | ResNet-34 | 14.98 | 0.10 |
| SMOKE[23] | DLA-34 | 12.85 | **0.03** |
| M3D-RPN[2] | DenseNet-121 | 17.06 | 0.16 |
| ImVoxelNet | ResNet-18 | 16.23 | 0.37 |
|  | ResNet-34 | 16.58 | 0.38 |
|  | ResNet-50 | **17.80** | 0.40 |

Table 7. $AP_{3D}$@0.7 for *car* category, *moderate* difficulty and runtime measured in seconds per image, estimated for the validation subset of the KITTI dataset.

## 5. Conclusion

In this paper, we formulate the task of multi-view RGB-based 3D object detection as an end-to-end optimization problem. To address this problem, we have proposed ImVoxelNet, a novel fully convolutional method of 3D object detection given posed monocular or multi-view RGB inputs. During both training and inference, ImVoxelNet accepts multi-view inputs with an arbitrary number of views. Besides, our method can accept monocular inputs (treated as a special case of multi-view inputs). The proposed method has achieved state-of-the-art results in outdoor car detection on both the monocular KITTI benchmark and the multi-view nuScenes benchmark. Moreover, it has surpassed existing methods of 3D object detection on the indoor SUN RGB-D dataset. For the ScanNet dataset, ImVoxelNet has set a new benchmark for indoor multi-view 3D object detection. Overall, ImVoxelNet successfully works on both indoor and outdoor data, which makes it general-purpose.

## References

[1] W. Bao, B. Xu, and Z. Chen. Monofenet: Monocular 3d object detection with feature enhancement networks. *IEEE Transactions on Image Processing*, 29:2753–2765, 2019.

[2] G. Brazil and X. Liu. M3d-rpn: Monocular 3d region proposal network for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9287–9296, 2019.

[3] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[4] F. Chabot, M. Chaouch, J. Rabarisoa, C. Teuliere, and T. Chateau. Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2040–2049, 2017.

[5] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu, et al. Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

[6] 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. Citeseer, 2015.

[7] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun. Monocular 3d object detection for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2147–2156, 2016.

[8] W. Choi, Y.-W. Chao, C. Pantofaru, and S. Savarese. Understanding indoor scenes using 3d geometric phrases. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 33–40, 2013.

[9] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5828–5839, 2017.

[10] M. Ding, Y. Huo, H. Yi, Z. Wang, J. Shi, Z. Lu, and P. Luo. Learning depth-guided convolutions for monocular 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 1000–1001, 2020.

[11] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[13] J. Hou, A. Dai, and M. Nießner. 3d-sis: 3d semantic instance segmentation of rgb-d scans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4421–4430, 2019.

[14] S. Huang, S. Qi, Y. Xiao, Y. Zhu, Y. N. Wu, and S.-C. Zhu. Cooperative holistic scene understanding: Unifying 3d object, layout, and camera pose estimation. *arXiv preprint arXiv:1810.13049*, 2018.

[15] S. Huang, S. Qi, Y. Zhu, Y. Xiao, Y. Xu, and S.-C. Zhu. Holistic 3d scene parsing and reconstruction from a single rgb image. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 187–203, 2018.

[16] S. Huang, Y. Chen, T. Yuan, S. Qi, Y. Zhu, and S.-C. Zhu. Perspectivenet: 3d object detection from a single rgb image via perspective points. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[17] M. Jaritz, J. Gu, and H. Su. Multi-view pointnet for 3d scene understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[18] A. Kundu, Y. Li, and J. M. Rehg. 3d-rcnn: Instance-level 3d object reconstruction via render-and-compare. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3559–3568, 2018.

[19] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019.

[20] B. Li, W. Ouyang, L. Sheng, X. Zeng, and X. Wang. Gs3d: An efficient 3d object detection framework for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1019–1028, 2019.

[21] P. Li, X. Chen, and S. Shen. Stereo r-cnn based 3d object detection for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7644–7652, 2019.

[22] P. Li, H. Zhao, P. Liu, and F. Cao. Rtm3d: Real-time monocular 3d detection from object keypoints for autonomous driving. *arXiv preprint arXiv:2001.03343*, 2, 2020.

[23] Z. Liu, Z. Wu, and R. Tóth. Smoke: single-stage monocular 3d object detection via keypoint estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 996–997, 2020.

[24] X. Ma, Z. Wang, H. Li, P. Zhang, W. Ouyang, and X. Fan. Accurate monocular 3d object detection via color-embedded 3d reconstruction for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6851–6860, 2019.

[25] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka. 3d bounding box estimation using deep learning and geometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7074–7082, 2017.

[26] Z. Murez, T. van As, J. Bartolozzi, A. Sinha, V. Badrinarayanan, and A. Rabinovich. Atlas: End-to-end 3d scene reconstruction from posed images. *arXiv preprint arXiv:2003.10432*, 2020.

[27] Y. Nie, X. Han, S. Guo, Y. Zheng, J. Chang, and J. J. Zhang. Total3dunderstanding: Joint layout, object pose and mesh reconstruction for indoor scenes from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 55–64, 2020.

[28] C. R. Qi, O. Litany, K. He, and L. J. Guibas. Deep hough voting for 3d object detection in point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9277–9286, 2019.

[29] C. R. Qi, X. Chen, O. Litany, and L. J. Guibas. Imvotenet: Boosting 3d object detection in point clouds with image votes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4404–4413, 2020.

[30] Z. Qin, J. Wang, and Y. Lu. Monogrnet: A geometric reasoning network for monocular 3d object localization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8851–8858, 2019.

[31] Z. Qin, J. Wang, and Y. Lu. Triangulation learning network: from monocular to stereo 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7615–7623, 2019.

[32] T. Roddick, A. Kendall, and R. Cipolla. Orthographic feature transform for monocular 3d object detection. *arXiv preprint arXiv:1811.08188*, 2018.

[33] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *European conference on computer vision*, pages 746–760. Springer, 2012.

[34] A. Simonelli, S. R. Bulo, L. Porzi, M. L. Antequera, and P. Kontschieder. Disentangling monocular 3d object detection: From single to multi-class recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[35] V. A. Sindagi, Y. Zhou, and O. Tuzel. Mvx-net: Multimodal voxelnet for 3d object detection. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7276–7282. IEEE, 2019.

[36] S. Song, S. P. Lichtenberg, and J. Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 567–576, 2015.

[37] Z. Tian, C. Shen, H. Chen, and T. He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9627–9636, 2019.

[38] Y. Yan, Y. Mao, and B. Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018.

[39] S. Zhang, C. Chi, Y. Yao, Z. Lei, and S. Z. Li. Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9759–9768, 2020.

[40] Z. Zhang, B. Sun, H. Yang, and Q. Huang. H3dnet: 3d object detection using hybrid geometric primitives. In *European Conference on Computer Vision*, pages 311–329. Springer, 2020.

[41] D. Zhou, J. Fang, X. Song, C. Guan, J. Yin, Y. Dai, and R. Yang. Iou loss for 2d/3d object detection. In *2019 International Conference on 3D Vision (3DV)*, pages 85–94. IEEE, 2019.