

# Learning Depth-Guided Convolutions for Monocular 3D Object Detection

Mingyu Ding<sup>1,2</sup> Yuqi Huo<sup>2,5</sup> Hongwei Yi<sup>3</sup> Zhe Wang<sup>4</sup> Jianping Shi<sup>4</sup> Zhiwu Lu<sup>2,5</sup> Ping Luo<sup>1</sup>

<sup>1</sup>The University of Hong Kong <sup>2</sup>Gaoling School of Artificial Intelligence, Renmin University of China

<sup>3</sup>Shenzhen Graduate School, Peking University <sup>4</sup>SenseTime Research

<sup>5</sup>Beijing Key Laboratory of Big Data Management and Analysis Methods, Beijing 100872, China

{myding, pluo}@cs.hku.hk {bohony, luzhiwu}@ruc.edu.cn

## Abstract

*3D object detection from a single image without LiDAR is a challenging task due to the lack of accurate depth information. Conventional 2D convolutions are unsuitable for this task because they fail to capture local object and its scale information, which are vital for 3D object detection. To better represent 3D structure, prior arts typically transform depth maps estimated from 2D images into a pseudo-LiDAR representation, and then apply existing 3D point-cloud based object detectors. However, their results depend heavily on the accuracy of the estimated depth maps, resulting in suboptimal performance. In this work, instead of using pseudo-LiDAR representation, we improve the fundamental 2D fully convolutions by proposing a new local convolutional network (LCN), termed Depth-guided Dynamic-Depthwise-Dilated LCN ( $D^4LCN$ ), where the filters and their receptive fields can be automatically learned from image-based depth maps, making different pixels of different images have different filters.  $D^4LCN$  overcomes the limitation of conventional 2D convolutions and narrows the gap between image representation and 3D point cloud representation. Extensive experiments show that  $D^4LCN$  outperforms existing works by large margins. For example, the relative improvement of  $D^4LCN$  against the state-of-the-art on KITTI is 9.1% in the moderate setting.  $D^4LCN$  ranks 1<sup>st</sup> on KITTI monocular 3D object detection benchmark at the time of submission (car, December 2019). The code is available at <https://github.com/dingmyu/D4LCN>.*

## 1. Introduction

3D object detection is a fundamental problem and has many applications such as autonomous driving and robotics. Previous methods show promising results by utilizing LiDAR device, which produces precise depth information in terms of 3D point clouds. However, due to the high-cost

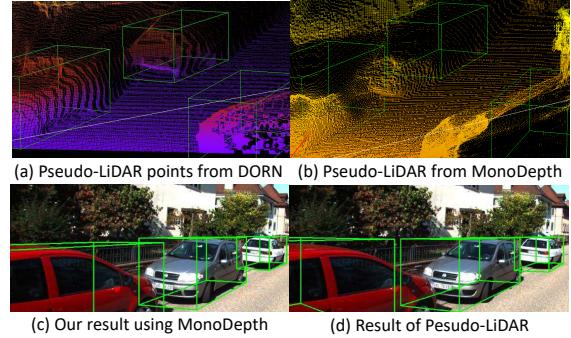


Figure 1. (a) and (b) show pseudo-LiDAR points generated by the supervised depth estimator, DORN [10] and the unsupervised Monodepth [13] respectively. The green box represents ground-truth (GT) 3D box. Pseudo-LiDAR points generated by inaccurate depth as shown in (b) have large offsets compared to the GT box. (c) and (d) show the detection results of our method and Pseudo-LiDAR [48] by using a coarse depth map. The performance of [48] depends heavily on the accuracy of the estimated depth maps, while our method achieves accurate detection results when accurate depth maps are missing.

and sparse output of LiDAR, it is desirable to seek cheaper alternatives like monocular cameras. This problem remains largely unsolved, though it has drawn much attention.

Recent methods towards the above goal can be generally categorized into two streams as image-based approaches [36, 26, 41, 19, 17, 4] and pseudo-LiDAR point-based approaches [48, 33, 50]. The image-based approaches [5, 17] typically leverage geometry constraints including object shape, ground plane, and key points. These constraints are formulated as different terms in loss function to improve detection results. The pseudo-LiDAR point-based approaches transform depth maps estimated from 2D images to point cloud representations to mimic the LiDAR signal. As shown in Figure 1, both of these methods have drawbacks, resulting in suboptimal performance.

Specifically, the image-based methods typically fail to capture meaningful local object scale and structure informa-

tion, because of the following two factors. (1) Due to perspective projection, the monocular view at far and near distance would cause significant changes in object scale. It is difficult for traditional 2D convolutional kernels to process objects of different scales simultaneously (see Figure 2). (2) The local neighborhood of 2D convolution is defined in the camera plane where the depth dimension is lost. In this non-metric space (*i.e.* the distance between pixels does not have a clear physical meaning like depth), a filter cannot distinguish objects from the background. In that case, a car area and the background area would be treated equally.

Although pseudo-LiDAR point-based approaches have achieved progressive results, they still possess two key issues. (1) The performance of these approaches heavily relies on the precision of estimated depth maps (see Figure 1). The depth maps extracted from monocular images are often coarse (point clouds estimated using them have wrong coordinates), leading to inaccurate 3D predictions. In other words, the accuracy of the depth map limits the performance of 3D object detection. (2) Pseudo-LiDAR methods cannot effectively employ high-level semantic information extracted from RGB images, leading to many false alarms. This is because point clouds provide spatial information but lose semantic information. As a result, regions like road-blocks, electrical boxes and even dust on the road may cause false detection, but they can be easily discriminated by using RGB images.

To address the above problems, we propose a novel convolutional network, termed Depth-guided Dynamic-Depthwise-Dilated local convolutional network ( $D^4LCN$ ), where the convolutional kernels are generated from the depth map and locally applied to each pixel and channel of individual image sample, rather than learning global kernels to apply to all images. As shown in Figure 2,  $D^4LCN$  treats the depth map as guidance to learn local dynamic-depthwise-dilated kernels from RGB images, so as to fill the gap between 2D and 3D representation. More specifically, the learned kernel in  $D^4LCN$  is sample-wise (*i.e.* exemplar kernel [15]), position-wise (*i.e.* local convolution [20]), and depthwise (*i.e.* depthwise convolution [18]), where each kernel has its own dilation rate (*i.e.* different exemplar kernels have different receptive fields).

$D^4LCN$  is carefully designed with four considerations. (1) The exemplar kernel is to learn specific scene geometry for each image. (2) The local convolution is to distinguish object and background regions for each pixel. (3) The depth-wise convolution is to learn different channel filters in a convolutional layer with different purposes and to reduce computational complexity. (4) The exemplar dilation rate is to learn different receptive fields for different filters to account for objects with diverse scales. The above delicate designs can be easily and efficiently implemented by combining linear operators of shift and element-wise product. As a

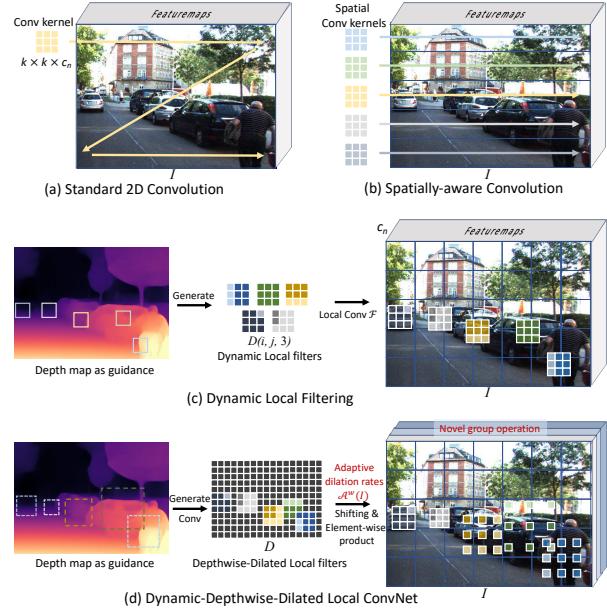


Figure 2. **Comparisons** among different convolutional approaches. (a) is the traditional 2D convolution that uses a single convolutional kernel applied on each pixel to convolve the entire image. (b) applies multiple fixed convolutional kernels on different regions (slices) of an image. (c) uses the depth map to generate dynamic kernels with the same receptive fields for each pixel. (d) denotes our approach, where the filter is dynamic, depth-wise, and has adaptive receptive fields for each pixel and channel of the feature map. It can be implemented more efficiently with fewer parameters than (c). Best viewed in color.

result, the efficient  $D^4LCN$  can not only address the problem of the scale-sensitive and meaningless local structure of 2D convolutions, but also benefit from the high-level semantic information from RGB images compared with the pseudo-LiDAR representation.

Our main **contributions** are three-fold. (1) A novel component for 3D object detection,  $D^4LCN$ , is proposed, where the depth map guides the learning of dynamic-depthwise-dilated local convolutions from a single monocular image. (2) We carefully design a single-stage 3D object detection framework based on  $D^4LCN$  to learn better 3D representation for reducing the gap between 2D convolutions and 3D point cloud-based operations. (3) Extensive experiments show that  $D^4LCN$  outperforms state-of-the-art monocular 3D detection methods and takes the first place on the KITTI benchmark [12].

## 2. Related Work

**Image-based Monocular 3D Detection.** Previous monocular 3D detection methods [36, 26, 41, 1, 19, 17, 4, 54] usually make assumptions about the scene geometry and use this as a constraint to train the 2D-to-3D mapping. Deep3DBox [36] uses the camera matrix to project a pre-

dicted 3D box onto the 2D image plane, constraining each side of the 2d detection box, such that it corresponds to any of the eight corners of the 3D box. OFTNet [43] introduces the orthographic feature transform, which maps image-based features into an orthographic 3D space. It is helpful when scale of objects varies drastically. [21, 31] investigated different ways of learning the confidence to model heteroscedastic uncertainty by using a 3D intersection-over-union (IoU) loss. To introduce more prior information, [2, 24, 57, 53] used 3D shapes as templates to get better object geometry. [23] predicts a point cloud in an object-centered coordinate system and devises a projection alignment loss to learn local scale and shape information. [34] proposes a 3D synthetic data augmentation algorithm via in-painting recovered meshes directly onto the 2D scenes.

However, as it is not easy for 2D image features to represent 3D structures, the above geometric constraints fail to restore accurate 3D information of objects from just a single monocular image. Therefore, our motivation is to utilize depth information, which essentially bridges gap between 2D and 3D representation, to guide learning the 2D-to-3D feature representation.

**Point Cloud-based Monocular 3D Detection.** Previous monocular methods [48, 33, 50] convert image-based depth maps to pseudo-LiDAR representations for mimicking the LiDAR signal. With this representation, existing LiDAR-based detection algorithms can be directly applied to monocular 3D object detection. For example, [50] detects 2D object proposals in the input image and extracts a point cloud frustum from the pseudo-LiDAR for each proposal. [33] proposes a multi-modal features fusion module to embed the complementary RGB cue into the generated point clouds representation. However, this depth-to-LiDAR transformation relies heavily on the accuracy of depth map and cannot make use of RGB information. In contrast, our method treats depth map as guidance to learn better 3D representation from RGB images.

**LiDAR-based 3D Detection.** With the development of deep learning on point sets, 3D feature learning [39, 40, 59] is able to learn deep point-based and voxel-based features. Benefit from this, LiDAR-based methods have achieved promising results in 3D detection. For example, [59] divides point clouds into equally spaced 3D voxels and transforms a group of points within each voxel into a unified feature representation. [47] applies the FPN technique to voxel-based detectors. [55] investigates a sparse convolution for voxel-based networks. [25] utilizes PointNets to learn a representation of point clouds organized in vertical columns (pillars). [38] leverages mature 2D object detectors to learn directly from 3D point clouds. [49] aggregates point-wise features as frustum-level feature vectors. [44, 7] directly generated a small number of high-quality 3D proposals from point clouds via segmenting the point clouds of

the whole scene into foreground and background. There are also some works focus on multi-sensor fusion (LiDAR as well as cameras) for 3D object detection. [29, 28] proposed a continuous fusion layer that encodes both discrete-state image features as well as continuous geometric information. [6, 22] used LiDAR point clouds and RGB images to generate features and encoded the sparse 3D point cloud with a compact multi-view representation.

**Dynamic Networks.** A number of existing techniques can be deployed to exploit the depth information for monocular 3D detection. M3D-RPN [1] proposes depth-aware convolution which uses non-shared kernels in the row-space to learn spatially-aware features. However, this rough and fixed spatial division has bias and fail to capture object scale and local structure. Dynamic filtering network [20] uses the sample-specific and position-specific filters but has heavy computational cost, and it also fails to solve the scale-sensitive problem of 2D convolutions. Trident network [27] utilizes manually defined multi-head detectors for 2D detection. However, it needs to manually group data for different heads. Other techniques like deformable convolution [8] and variants of [20] such as [14, 46, 52, 11], fail to capture object scale and local structure as well. In this work, our depth-guided dynamic dilated local convolutional network is proposed to solve the two problems associated with 2D convolutions and narrow the gap between 2D convolution and point cloud-based 3D processing.

### 3. Methodology

As a single-stage 3D detector, our framework consists of three key components: a network backbone, a depth-guided filtering module, and a 2D-3D detection head (see Figure 3). Details of each component are given below. First, we give an overview of our architecture as well as backbone networks. We then detail our depth-guided filtering module which is the key component for bridging 2D convolutions and the point cloud-based 3D processing. Finally, we outline the details of our 2D-3D detection head.

#### 3.1. Backbone

To utilize depth maps as guidance of 2D convolutions, we formulate our backbone as a two-branch network: the first branch is the feature extraction network using RGB images, and the other is the filter generation network to generate convolutional kernels for feature extraction network using the estimated depth as input. These two networks process the two inputs separately and their outputs of each block are merged by the depth-guided filtering module.

The backbone of the feature extraction network is ResNet-50 [16] without its final FC and pooling layers, and is pre-trained on the ImageNet classification dataset [9]. To obtain a larger field-of-view and keep the network stride at 16, we find the last convolutional layer (conv5.1, block4)

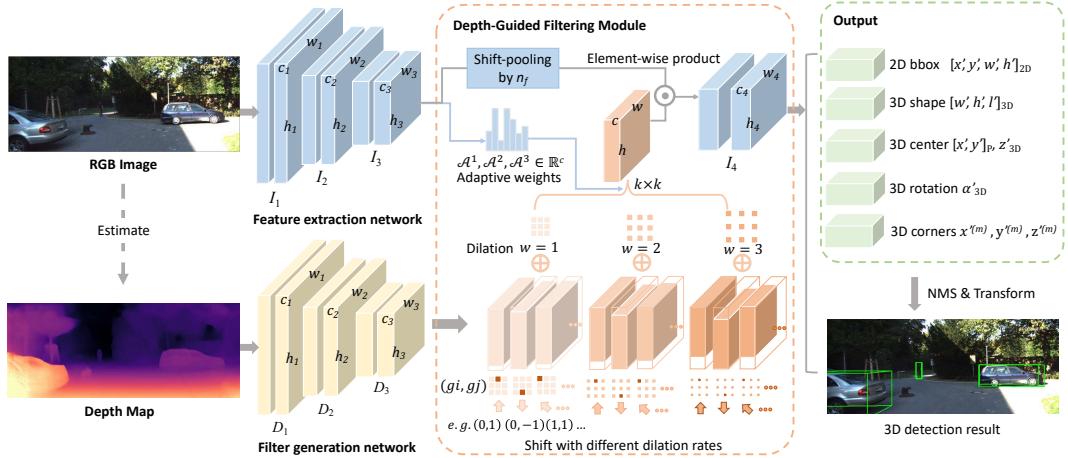


Figure 3. Overview of our framework for monocular 3D object detection. The depth map is first estimated from the RGB image and used as the input of our two-branch network together with the RGB image. Then the depth-guided filtering module is used to fuse these two information of each residual block. Finally, a one-stage detection head with Non-Maximum Suppression (NMS) is employed for prediction.

that decreases resolution and set its stride to 1 to avoid signal decimation, and replace all subsequent convolutional layers with dilated convolutional layers (the dilation rate is 2). For the filter generation network, we only use the first three blocks of ResNet-50 to reduce computational costs. Note the two branches have the same number of channels of each block for the depth guided filtering module.

### 3.2. Depth-Guided Filtering Module

Traditional 2D convolution kernels fail to efficiently model the depth-dependent scale variance of the objects and effectively reason about the spatial relationship between foreground and background pixels. On the other hand, pseudo-lidar representations rely too much on the accuracy of depth and lose the RGB information. To address these problems simultaneously, we propose our depth-guided filtering module. Notably, by using our module, the convolutional kernels and their receptive fields (dilation) are different for different pixels and channels of different images.

Since the kernel of our feature extraction network is trained and generated by the depth map, it is sample-specific and position-specific, as in [20, 14], and thus can capture meaningful local structures as the point-based operator in point clouds. We first introduce the idea of depth-wise convolution [18] to the network, termed depth-wise local convolution (DLCN). Generally, depth-wise convolution (DCN) involves a set of global filters, where each filter only operates at its corresponding channel, while DLCN requires a feature volume of local filters the same size as the input feature maps. As the generated filters are actually a feature volume, a naive way to perform DLCN requires to convert the feature volume into  $h_n \times w_n$  location-specific filters and then apply depth-wise and local convolutions to the feature maps, where  $h_n$  and  $w_n$  are the height and width of the feature maps at layer  $n$ . This implementation

would be time-consuming as it ignores the redundant computations in neighboring pixels. To reduce the time cost, we employ the shift and element-wise product operators, in which shift [51] is a zero-flop zero-parameter operation, and element-wise product requires little calculation. Concretely, let  $I_n \in \mathbb{R}^{h_n \times w_n \times c_n}$  and  $D_n \in \mathbb{R}^{h_n \times w_n \times c_n}$  be the output of the feature extraction network and filter generation network, respectively, where  $n$  is the index of the block (note that block  $n$  corresponds to the layer  $conv_{n+1}$  in ResNet). Let  $k$  denote the kernel size of the feature extraction network. By defining a shifting grid  $\{(g_i, g_j)\}$ ,  $g \in (int)[1 - k/2, k/2 - 1]$  that contains  $k \cdot k$  elements, for every vector  $(g_i, g_j)$ , we shift the whole feature map  $I \odot D$  towards the direction and step size indicated by  $(g_i, g_j)$  and get the result  $(I \odot D)^{(g_i, g_j)}$ . For example,  $g \in \{-1, 0, 1\}$  when  $k = 3$ , and the feature map is moved towards nine directions with a horizontal or vertical step size of 0 or 1. We then use the sum and element-wise product operations to compute our filtering result:

$$I' = \frac{1}{k \cdot k} \sum_{g_i, g_j} (I \odot D)^{(g_i, g_j)}. \quad (1)$$

To encourage information flow between channels of the depth-wise convolution, we further introduce a novel shift-pooling operator in the module. Considering  $n_f$  as the number of channels with information flow, we shift the feature maps along the channel axis for  $n_f$  times by 1, 2, ...,  $n_f - 1$  to obtain new  $n_f - 1$  shifted feature maps  $I_s^{(n_i)}$ ,  $n_i \in \{1, 2, \dots, n_f - 1\}$ . Then we perform element-wise mean to the shifted feature maps and the original  $I$  to obtain the new feature map as the input of the module. The process of this shift-pooling operation is shown in Figure 4 ( $n_f = 3$ ).

Compared to the idea ‘group’ of depth-wise convolution in [18, 58] which aims to group many channels into a group

to perform information fusion between them, the proposed shift-pooling operator is more efficient and adds no additional parameters to the convolution. The size of our convolutional weights of each local kernel is always  $k \times k \times c_n$  when applying shift-pooling, while it changes significantly in [18] for different number of groups from  $k \times k \times c_n$  to  $k \times k \times c_n \times c_n$  in group convolution (assume that the convolution keeps the number of channels unchanged). Note that it is difficult for the filter generation network to generate so many kernels for the traditional convolutions  $\mathcal{F}$  between all channels, and the characteristic of being position-specific dramatically increases their computational cost.

With our depth-wise formulation, different kernels can have different functions. This enables us to assign different dilation rates [56] for each filter to address the scale-sensitive problem. Since there are huge intra-class and inter-class scale differences in an RGB image, we use  $I$  to learn an adaptive dilation rate for each filter to obtain different sizes of receptive fields by an adaptive function  $\mathcal{A}$ . Specifically, let  $d$  denote our maximum dilation rate, the adaptive function  $\mathcal{A}$  consists of three layers: (1) an AdaptiveMaxPool2d layer with the output size of  $d \times d$  and channel number  $c$ ; (2) a convolutional layer with a kernel size of  $d \times d$  and channel number  $d \times c$ ; (3) a reshape and softmax layer to generate  $d$  weights  $\mathcal{A}^w(I)$ ,  $w \in (\text{int}[1, d]$  with a sum of 1 for each filter. Formally, our guided filtering with adaptive dilated function ( $D^4$ LCN) is formulated as follows:

$$I' = \frac{1}{d \cdot k \cdot k} \cdot \sum_w \mathcal{A}^w(I) \sum_{g_i, g_j} (I \odot D)^{(g_i * w, g_j * w)}, \quad (2)$$

For different images, our depth-guided filtering module assigns different kernels on different pixels and adaptive receptive fields (dilation) on different channels. This solves the problem of scale-sensitive and meaningless local structure of 2D convolutions, and also makes full use of RGB information compared to pseudo-LiDAR representations.

### 3.3. 2D-3D Detection Head

In this work, we adopt a single-stage detector with prior-based 2D-3D anchor boxes [42, 32] as our base detector.

#### 3.3.1 Formulation

**Inputs:** The output feature map  $I_4 \in \mathbb{R}^{h_4 \times w_4}$  of our backbone network with a network stride factor of 16. Following common practice, we use a calibrated setting which assumes that per-image camera intrinsics  $K \in \mathbb{R}^{3 \times 4}$  are available both at the training and test time. The 3D-to-2D projection can be written as:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}_P \cdot z_{3D} = K \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_{3D} \quad (3)$$

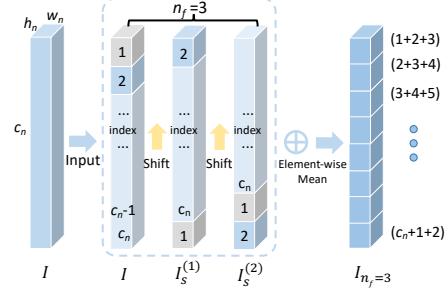


Figure 4. An example of our shift-pooling operator of depth-wise convolution in depth-guided filtering module when  $n_f$  is 3. It is efficiently implemented by shift and element-wise mean operators.

where  $[x, y, z]_{3D}$  denotes the horizontal position, height and depth of the 3D point in camera coordinates, and  $[x, y]_P$  is the projection of the 3D point in 2D image coordinates.

**Ground Truth:** We define a ground truth (GT) box using the following parameters: the 2D bounding box  $[x, y, w, h]_{2D}$ , where  $(x, y)$  is the center of 2D box and  $w, h$  are the width and height of 2D box; the 3D center  $[x, y, z]_{3D}$  represents the location of 3D center in camera coordinates; the 3D shapes  $[w, h, l]_{3D}$  (3D object dimensions: height, width, length (in meters)), and the allocentric pose  $\alpha_{3D}$  in 3D space (observation angle of object, ranging  $[-\pi, \pi]$ ) [34]. Note that we use the minimum enclosing rectangle of the projected 3D box as our ground truth 2D bounding box.

**Outputs:** Let  $n_a$  denote the number of anchors and  $n_c$  denote the number of classes. For each position  $(i, j)$  of the input, the output for an anchor contains  $35 + n_c$  parameters:  $\{[t_x, t_y, t_w, t_h]_{2D}, [t_x, t_y]_P, [t_z, t_w, t_h, t_l, t_\alpha]_{3D}, t_C^{(m)}, s\}$ , where  $[t_x, t_y, t_w, t_h]_{2D}$  is the predicted 2D box;  $[t_x, t_y]_P$  is the position of the projected 3D corner in the 2D plane,  $[t_z, t_w, t_h, t_l, t_\alpha]_{3D}$  denotes the depth, predicted 3D shape and rotation, respectively;  $t_C^{(m)} = \{[t_x^{(m)}, t_y^{(m)}]_P, [t_z^{(m)}]_{3D}\}$ ,  $m \in \{1, 2, \dots, 8\}$  denotes 8 projected 3D corners;  $s$  denotes the classification score of each class. The size of the output is  $h_4 \times w_4 \times n_a \times (35 + n_c)$ , where  $(h_4, w_4)$  is the size of the input image with a down sampling factor of 16. The output is actually an anchor-based transformation of the 2D-3D box.

#### 3.3.2 2D-3D Anchor

Inspired by [1], we utilize 2D-3D anchors with priors as our default anchor boxes. More specifically, a 2D-3D anchor is first defined on the 2D space as in [32] and then use the corresponding priors in the training dataset to calculate the part of it in the 3D space. One template anchor is defined using parameters of both spaces:  $\{[A_x, A_y, A_w, A_h]_{2D}, [A_z, A_w, A_h, A_l, A_\alpha]_{3D}\}$ , where  $[A_z, A_w, A_h, A_l, A_\alpha]_{3D}$  denotes the 3D anchor (depth, shape, rotation).

For 2D anchors  $[A_x, A_y, A_w, A_h]_{2D}$ , we use 12 differ-

ent scales ranging from 30 to 400 pixels in height following the power function of  $30 * 1.265^{\exp}$ ,  $\exp \in (\text{int})[0, 11]$  and aspect ratios of  $[0.5, 1.0, 1.5]$  to define a total of 36 anchors. We then project all ground truth 3D boxes to the 2D space. For each projected box, we calculate its intersection over union (IoU) with each 2D anchor and assign the corresponding 3D box to the anchors that have an IoU  $\geq 0.5$ . For each 2D anchor, we thus use the statistics across all matching ground truth 3D boxes as its corresponding 3D anchor  $[A_z, A_w, A_h, A_l, A_\alpha]_{3D}$ . Note that we use the same anchor parameters  $[A_x, A_y]_{2D}$  for the regression of  $[t_x, t_y]_{2D}$  and  $[t_x, t_y]_P$ . The anchors enable our network to learn a relative value (residual) of the ground truth, which significantly reduces the difficulty of learning.

### 3.3.3 Data Transformation

We combine the output of our network which is an anchor-based transformation of the 2D-3D box and the pre-defined anchors to obtain our estimated 3D boxes:

$$\begin{aligned} [x', y']_{2D} &= [A_x, A_y]_{2D} + [t_x, t_y]_{2D} * [A_w, A_h]_{2D} \\ [x', y']_P &= [A_x, A_y]_{2D} + [t_x, t_y]_P * [A_w, A_h]_{2D} \\ [x'^{(m)}, y'^{(m)}]_P &= [A_x, A_y]_{2D} + [t_x^{(m)}, t_y^{(m)}]_P * [A_w, A_h]_{2D} \\ [w', h']_{2D} &= [A_w, A_h]_{2D} \cdot \exp([t_w, t_h]_{2D}) \\ [w', h', l']_{3D} &= [A_w, A_h, A_l]_{3D} \cdot \exp([t_w, t_h, t_l]_{3D}) \\ [z', z'^{(m)}, \alpha']_{3D} &= [A_z, A_z, A_\alpha] + [t_z, t_z, t_{alpha}]_{3D}. \end{aligned} \quad (4)$$

where  $[x', y']_P, [z', z'^{(m)}, \alpha']_{3D}$  denote respectively the estimated 3D center projection in 2D plane, the depth of 3D center and eight corners, the 3D rotation by combining output of the network and the anchor.

### 3.3.4 Losses

Our overall loss contains a classification loss, a 2D regression loss, a 3D regression loss and a 2D-3D corner loss. We use the idea of focal loss [30] to balance the samples. Let  $s_t$  and  $\gamma$  denote the classification score of target class and the focusing parameter, respectively. We have:

$$L = (1 - s_t)^\gamma (L_{class} + L_{2d} + L_{3d} + L_{corner}), \quad (5)$$

where  $\gamma = 0.5$  in all experiments, and  $L_{class}$ ,  $L_{2d}$ ,  $L_{3d}$ ,  $L_{corner}$  are the classification loss, 2D regression loss, 3D regression loss and D-3D corner loss, respectively.

In this work, we employ the standard cross-entropy (CE) loss for classification:

$$L_{class} = -\log(s_t). \quad (6)$$

Moreover, for both 2D and 3D regression, we simply use

the SmoothL1 regression losses:

$$\begin{aligned} L_{2D} &= \text{SmoothL1}([x', y', w', h']_{2D}, [x, y, w, h]_{2D}), \\ L_{3D} &= \text{SmoothL1}([w', h', l', z', \alpha']_{3D}, [w, h, l, z, \alpha]_{3D}), \\ &+ \text{SmoothL1}([x', y']_P, [x, y]_P), \\ L_{corner} &= \frac{1}{8} \sum \text{SmoothL1}([x'^{(m)}, y'^{(m)}]_P, [x^{(m)}, y^{(m)}]_P) \\ &+ \text{SmoothL1}([z'^{(m)}]_{3D}, [z]_{3D}), \end{aligned} \quad (7)$$

where  $[x^{(m)}, y^{(m)}]_P$  denotes the projected corners in image coordinates of the GT 3D box and  $[z]_{3D}$  is its GT depth.

## 4. Experiments

### 4.1. Dataset and Setting

**KITTI Dataset.** The KITTI 3D object detection dataset [12] is widely used for monocular and LiDAR-based 3D detection. It consists of 7,481 training images and 7,518 test images as well as the corresponding point clouds and the calibration parameters, comprising a total of 80,256 2D-3D labeled objects with three object classes: Car, Pedestrian, and Cyclist. Each 3D ground truth box is assigned to one out of three difficulty classes (easy, moderate, hard) according to the occlusion and truncation levels of objects. There are two train-val splits of KITTI: the split1 [5] contains 3,712 training and 3,769 validation images, while the split2 [53] uses 3,682 images for training and 3,799 images for validation. The dataset includes three tasks: 2D detection, 3D detection, and Bird’s eye view, among which 3D detection is the focus of 3D detection methods.

**Evaluation Metrics.** Precision-recall curves are used for evaluation (with the IoU threshold of 0.7). Prior to Aug. 2019, 11-point Interpolated Average Precision (AP) metric  $AP|_{R_{11}}$  proposed in the Pascal VOC benchmark is separately computed on each difficulty class and each object class. After that, the 40 recall positions-based metric  $AP|_{R_{40}}$  is used instead of  $AP|_{R_{11}}$ , following [45]. All methods are ranked by  $AP|_{R_{11}}$  of the 3D car detection in the moderate setting.

**Implementation Details.** We use our depth-guided filtering module three times on the first three blocks of ResNet, which have different network strides of 4,8,16, respectively. [10] is used for depth estimation. A drop-channel layer with a drop rate of 0.2 is used after each module and a dropout layer with a drop rate of 0.5 is used after the output of the network backbone. For our single-stage detector, we use two convolutional layers as our detection head. The number of channels in the first layer is 512, and  $n_a * (35 + n_c)$  for the second layer, where  $n_c$  is set to 4 for three object classes and the background class, and  $n_a$  is set to 36. Non Maximum Suppression (NMS) with an IoU threshold of 0.4 is used on the network output in 2D space. Since the regression of the 3D rotation  $\alpha$  is more difficult than other parameters, a hill-climbing post-processing step is used for optimizing  $\alpha$  as in

Method	Test set			Split1			Split2		
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
OFT-Net [43]	1.61	1.32	1.00	4.07	3.27	3.29	–	–	–
FQNet [31]	2.77	1.51	1.01	5.98	5.50	4.75	5.45	5.11	4.45
ROI-10D [34]	4.32	2.02	1.46	10.25	6.39	6.18	–	–	–
GS3D [26]	4.47	2.90	2.47	13.46	10.97	10.38	11.63	10.51	10.51
Shift R-CNN [37]	6.88	3.87	2.83	13.84	11.29	11.08	–	–	–
MonoGRNet [41]	9.61	5.74	4.25	13.88	10.19	7.62	–	–	–
MonoPSR [23]	10.76	7.25	5.85	12.75	11.48	8.59	13.94	12.24	10.77
Mono3D-PLiDAR [50]	10.76	7.50	6.10	<b>31.50</b>	21.00	17.50	–	–	–
SS3D [21]	10.78	7.68	6.51	14.52	13.15	11.85	9.45	8.42	7.34
MonoDIS [45]	10.37	7.94	6.40	11.06	7.60	6.37	–	–	–
Pseudo-LiDAR [48]	–	–	–	19.50	17.20	16.20	–	–	–
M3D-RPN [1]	14.76	9.71	7.42	20.27	17.06	15.21	<b>20.40</b>	<b>16.48</b>	<b>13.34</b>
AM3D [1]	<b>16.50</b>	<b>10.74</b>	<b>9.52 (+0.01)</b>	<b>32.23 (+5.26)</b>	<b>21.09</b>	<b>17.26</b>	–	–	–
<b>D<sup>4</sup>LCN (Ours)</b>	<b>16.65 (+0.15)</b>	<b>11.72 (+0.98)</b>	<b>9.51</b>	26.97	<b>21.71 (+0.62)</b>	<b>18.22 (+0.96)</b>	<b>24.29 (+3.89)</b>	<b>19.54 (+3.06)</b>	<b>16.38 (+3.04)</b>

Table 1. Comparative results on the KITTI 3D object detection dataset. For the test set, only  $AP|_{R_{40}}$  is provided by the official leaderboard. We thus show the results on the test set in  $AP|_{R_{40}}$  and split1/split2 in  $AP|_{R_{11}}$ . We use **red** to indicate the highest result with relative improvement in parentheses and **blue** for the second-highest result of the class *car*. Our method achieves 7 firsts and 2 seconds in 9 items.

[1]. The input images are scaled to  $512 \times 1760$  and horizontal flipping is the only data augmentation.  $n_f$  is set to 3 and the maximum dilation rate  $d$  is set to 3 in all experiments.

The network is optimized by stochastic gradient descent (SGD), with a momentum of 0.9 and a weight decay of 0.0005. We take a mini-batch size of 8 on 4 Nvidia Tesla v100 GPUs (16G). We use the ‘poly’ learning rate policy and set the base learning rate to 0.01 and power to 0.9. The iteration number for the training process is set to 40,000.

## 4.2. Comparative Results

We conduct experiments on the official test set and two splits of validation set of the KITTI dataset. Table 1 includes the top 14 monocular methods in the leaderboard, among which our method ranks top-1. We can observe that: (1) Our method outperforms the second-best competitor for monocular 3D car detection by a large margin (relatively 9.1% for 10.74 vs. 11.72) under the moderate setting (which is the most important setting of KITTI). (2) Most competitors, such as [23, 33, 45, 37, 50, 1], utilize the detector (e.g. Faster-RCNN) pre-trained on COCO/KITTI or resort to multi-stage training to obtain better 2D detection and stable 3D results, while our model is trained end-to-end using the standard ImageNet pre-trained model. However, we still achieve the state-of-the-art 3D detection results, validating the effectiveness of our D<sup>4</sup>LCN to learn 3D structure. (3) Recently KITTI uses  $AP|_{R_{40}}$  instead of  $AP|_{R_{11}}$ , however, all existing methods report the results under the old metric. We thus also give results under  $AP|_{R_{11}}$  on the validation set for fair comparison. It can be seen that our method outperforms all others on the two splits for 3D car detection. Our results under  $AP|_{R_{40}}$  on validation set are shown in ablation study.

Method	Task	AP  <sub>R<sub>11</sub></sub>			AP  <sub>R<sub>40</sub></sub>		
		Easy	Moderate	Hard	Easy	Moderate	Hard
3DNet	2D detection	93.42	85.16	68.14	94.13	84.45	65.73
	<b>3D detection</b>	17.94	14.61	12.74	16.72	12.13	09.46
	Bird’s-eye view	24.87	19.89	16.14	23.19	16.67	13.39
+CL	2D detection	<b>94.04</b>	<b>85.56</b>	68.50	<b>94.98</b>	84.93	66.11
	<b>3D detection</b>	20.66	15.57	13.41	17.10	12.09	09.47
	Bird’s-eye view	2903	23.82	19.41	24.12	17.75	13.66
+DLCN	2D detection	92.98	85.35	68.63	93.81	86.71	70.19
	<b>3D detection</b>	23.25	17.92	15.58	18.32	13.50	10.61
	Bird’s-eye view	27.76	22.89	18.73	26.78	18.68	15.14
+SP	2D detection	92.57	85.14	68.40	93.35	86.52	67.93
	<b>3D detection</b>	25.30	19.02	17.26	19.69	14.44	11.52
	Bird’s-eye view	31.39	24.40	19.85	26.91	20.07	15.77
D <sup>4</sup> LCN	2D detection	93.59	85.51	<b>68.81</b>	94.25	<b>86.93</b>	<b>70.34</b>
	<b>3D detection</b>	<b>26.97</b>	<b>21.71</b>	<b>18.22</b>	<b>22.32</b>	<b>16.20</b>	<b>12.30</b>
	Bird’s-eye view	<b>34.82</b>	<b>25.83</b>	<b>23.53</b>	<b>31.53</b>	<b>22.58</b>	<b>17.87</b>

Table 2. Ablation study on the class *car* on the KITTI split1.

## 4.3. Detailed Analysis

### 4.3.1 Ablation Study

To conduct ablation study on our model, we make comparison among five versions of our model: (1) 3DNet: the baseline model using  $L_{2D}$  and  $L_{3D}$  without our depth-guided filtering module; (2) + CL: the Corner Loss is added to 3DNet; (3) + DLCN: depth-guided depth-wise local filtering is added; (4) + SP: shift-pooling operator is added (with  $n_f = 3$ ); (5) D<sup>4</sup>LCN (our full model): adaptive dilation rates are added, as in Eq. 2. From Table 2, we can observe that: (1) The performance continuously increases when more components are used for 3D object detection, showing the contribution of each component. (2) Our depth-guided filtering module increases the 3D detection AP scores (moderate) from  $\{15.57, 12.09\}$  to  $\{21.71, 16.20\}$  w.r.t. the  $AP|_{R_{11}}$  and  $AP|_{R_{40}}$  metrics, respectively. This suggests that it is indeed effective to capture the meaningful local structure for 3D object detection. (3) The main improvement comes from our adaptive dilated convolution (2.69 and 1.76

Depth	AP  <sub>R<sub>11</sub></sub>			AP  <sub>R<sub>40</sub></sub>		
	Easy	Moderate	Hard	Easy	Moderate	Hard
MonoDepth [13]	22.43	19.63	16.38	16.82	13.18	10.87
DORN [10]	26.97	21.71	18.22	22.32	16.20	12.30
DispNet [35]	<b>30.95</b>	24.06	20.29	<b>25.73</b>	18.56	15.10
PSMNet [3]	30.03	<b>25.41</b>	<b>21.63</b>	25.24	<b>19.80</b>	<b>16.45</b>

Table 3. Comparisons of depth maps of different quality for 3D detection on the class *car* on the KITTI split1.

Class	Easy		Moderate		Hard	
	[split1/split2/test]	[split1/split2/test]	[split1/split2/test]	[split1/split2/test]	[split1/split2/test]	[split1/split2/test]
Car	26.97/24.29/16.65		21.71/19.54/11.72		18.22/16.38/9.51	
Pedestrian	12.95/12.52/4.55		11.23/10.37/3.42		11.05/10.23/2.83	
Cyclist	5.85/7.05/2.45		4.41/6.54/1.67		4.14/6.54/1.36	

Table 4. Multi-class 3D detection results of our method on the three data splits. Note that all pseudo-LiDAR based methods [33, 50, 48] fail to detect pedestrians and cyclists.

for  $AP|_{R_{11}}$  and  $AP|_{R_{40}}$ , respectively), which allows each channel of the feature map to have different receptive fields and thus solves the scale-sensitive problem. Note that we have tried different values of  $n_f \in \{1, 2, 3, 4, 5, 6\}$ , and found that  $n_f = 3$  is the best.

### 4.3.2 Evaluation of Depth Maps

To study the impact of accuracy of depth maps on the performance of our method, we extract depth maps using four different methods [13, 10, 35, 3] and then apply them to 3D detection. As reported in previous works on depth estimation, the three supervised methods (*i.e.* PSMNet, DispNet, and DORN) significantly outperform the unsupervised method [13]. Among the supervised methods, Stereo-based methods [3, 35] are better than monocular-based DORN. With these conclusions, we have the following observations from Table 3: (1) The accuracy of 3D detection is higher with better depth map. This is because that better depth map can provide better scene geometry and local structure. (2) As the quality of depth map increases, the growth of detection accuracy becomes slower. (3) Even with the depth maps obtained by unsupervised learning [13], our method achieves state-of-the-art results. Compared to the pseudo-LiDAR based method [33], our method relies less on the quality of depth maps (19.63 vs. 15.45 using MonoDepth).

### 4.3.3 Multi-Class 3D Detection

Since a person is a non-rigid body, its shape varies and its depth information is hard to accurately estimate. For this reason, 3D detection of pedestrians and cyclists becomes particularly difficult. Note that all pseudo-LiDAR based methods [33, 50, 48] fail to detect these two categories. However, as shown in Table 4, our method still achieves satisfactory performance on 3D detection of pedestrians and



Figure 5. Visualization of active maps corresponding to different filters of block 3 of our  $D^4LCN$ . Each filter learns three weights representing dilation rate of 1, 2, 3, respectively. Different filters have different functions in our model to handle the scale problem adaptively. For example, filter 89 has large receptive fields for large-scale cars, while filter 70 deals with the small-scale cars.

cyclists. Moreover, we also show the active maps corresponding to different filters of our  $D^4LCN$  in Figure 5. Different filters on the same layer of our model use different sizes of receptive fields to handle objects of different scales, including pedestrians (small) and cars (big), as well as distant cars (big) and nearby cars (small).

## 5. Conclusion

In this paper, we propose a Depth-guided Dynamic-Depthwise-Dilated Local ConvNet ( $D^4LCN$ ) for monocular 3D objection detection, where the convolutional kernels and their receptive fields (dilation rates) are different for different pixels and channels of different images. These kernels are generated dynamically conditioned on the depth map to compensate the limitations of 2D convolution and narrow the gap between 2D convolutions and the point cloud-based 3D operators. As a result, our  $D^4LCN$  can not only address the problem of the scale-sensitive and meaningless local structure of 2D convolutions, but also benefit from the high-level semantic information from RGB images. Extensive experiments show that our  $D^4LCN$  better captures 3D information and ranks 1<sup>st</sup> for monocular 3D object detection on the KITTI dataset at the time of submission.

## 6. Acknowledgements

This work was partially supported by HKU Seed Fund for Basic Research, Start-up Fund and Research Donation from SenseTime. Zhiwu Lu is partially supported by National Natural Science Foundation of China (61976220, 61832017, and 61573363), and Beijing Outstanding Young Scientist Program (BJWZYJH012019100020098). Thank Dr. Guorun Yang for his careful proofreading.

## References

- [1] Garrick Brazil and Xiaoming Liu. M3d-rpn: Monocular 3d region proposal network for object detection. In *ICCV*, pages 9287–9296, 2019. [2](#), [3](#), [5](#), [7](#)
- [2] Florian Chabot, Mohamed Chaouch, Jaonary Rabarisoa, Céline Teulière, and Thierry Chateau. Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image. In *CVPR*, pages 2040–2049, 2017. [3](#)
- [3] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *CVPR*, pages 5410–5418, 2018. [8](#)
- [4] Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. Monocular 3d object detection for autonomous driving. In *CVPR*, pages 2147–2156, 2016. [1](#), [2](#)
- [5] Xiaozhi Chen, Kaustav Kundu, Yukun Zhu, Andrew G Berneshawi, Huimin Ma, Sanja Fidler, and Raquel Urtasun. 3d object proposals for accurate object class detection. In *NeurIPS*, pages 424–432, 2015. [1](#), [6](#)
- [6] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *CVPR*, pages 1907–1915, 2017. [3](#)
- [7] Yilun Chen, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Fast point r-cnn. In *ICCV*, 2019. [3](#)
- [8] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *ICCV*, pages 764–773, 2017. [3](#)
- [9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009. [3](#)
- [10] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *CVPR*, pages 2002–2011, 2018. [1](#), [6](#), [8](#)
- [11] Peng Gao, Hongsheng Li, Shuang Li, Pan Lu, Yikang Li, Steven CH Hoi, and Xiaogang Wang. Question-guided hybrid convolution for visual question answering. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 469–485, 2018. [3](#)
- [12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, pages 3354–3361, 2012. [2](#), [6](#)
- [13] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, 2017. [1](#), [8](#)
- [14] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016. [3](#), [4](#)
- [15] Kaiming He, Jian Sun, and Xiaoou Tang. Guided image filtering. In *ECCV*, pages 1–14, 2010. [2](#)
- [16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. [3](#)
- [17] Tong He and Stefano Soatto. Mono3d++: Monocular 3d vehicle detection with two-scale 3d hypotheses and task priors. *arXiv preprint arXiv:1901.03446*, 2019. [1](#), [2](#)
- [18] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. [2](#), [4](#), [5](#)
- [19] Hou-Ning Hu, Qi-Zhi Cai, Dequan Wang, Ji Lin, Min Sun, Philipp Krahenbuhl, Trevor Darrell, and Fisher Yu. Joint monocular 3d vehicle detection and tracking. In *ICCV*, pages 5390–5399, 2019. [1](#), [2](#)
- [20] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. In *NeurIPS*, pages 667–675, 2016. [2](#), [3](#), [4](#)
- [21] Eskil Jørgensen, Christopher Zach, and Fredrik Kahl. Monocular 3d object detection and box fitting trained end-to-end using intersection-over-union loss. *arXiv preprint arXiv:1906.08070*, 2019. [3](#), [7](#)
- [22] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L Waslander. Joint 3d proposal generation and object detection from view aggregation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8, 2018. [3](#)
- [23] Jason Ku, Alex D Pon, and Steven L Waslander. Monocular 3d object detection leveraging accurate proposals and shape reconstruction. In *CVPR*, pages 11867–11876, 2019. [3](#), [7](#)
- [24] Abhijit Kundu, Yin Li, and James M Rehg. 3d-rcnn: Instance-level 3d object reconstruction via render-and-compare. In *CVPR*, pages 3559–3568, 2018. [3](#)
- [25] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, pages 12697–12705, 2019. [3](#)
- [26] Buyu Li, Wanli Ouyang, Lu Sheng, Xingyu Zeng, and Xiaogang Wang. Gs3d: An efficient 3d object detection framework for autonomous driving. In *CVPR*, pages 1019–1028, 2019. [1](#), [2](#), [7](#)
- [27] Yanghao Li, Yuntao Chen, Naiyan Wang, and Zhaoxiang Zhang. Scale-aware trident networks for object detection. In *ICCV*, 2019. [3](#)
- [28] Ming Liang, Bin Yang, Yun Chen, Rui Hu, and Raquel Urtasun. Multi-task multi-sensor fusion for 3d object detection. In *CVPR*, pages 7345–7353, 2019. [3](#)
- [29] Ming Liang, Bin Yang, Shenlong Wang, and Raquel Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *ECCV*, pages 641–656, 2018. [3](#)
- [30] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, pages 2980–2988, 2017. [6](#)
- [31] Lijie Liu, Jiwen Lu, Chunjing Xu, Qi Tian, and Jie Zhou. Deep fitting degree scoring network for monocular 3d object detection. In *CVPR*, pages 1057–1066, 2019. [3](#), [7](#)
- [32] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *ECCV*, pages 21–37, 2016. [5](#)
- [33] Xinzhu Ma, Zhihui Wang, Haojie Li, Pengbo Zhang, Wanli Ouyang, and Xin Fan. Accurate monocular 3d object detection via color-embedded 3d reconstruction for autonomous driving. In *ICCV*, pages 6851–6860, 2019. [1](#), [3](#), [7](#), [8](#)

- [34] Fabian Manhardt, Wadim Kehl, and Adrien Gaidon. Roi-10d: Monocular lifting of 2d detection to 6d pose and metric shape. In *CVPR*, pages 2069–2078, 2019. 3, 5, 7
- [35] Nikolas Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *CVPR*, pages 4040–4048, 2016. 8
- [36] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka. 3d bounding box estimation using deep learning and geometry. In *CVPR*, pages 7074–7082, 2017. 1, 2
- [37] Andretti Naiden, Vlad Paunescu, Gyeongmo Kim, Byeong-Moon Jeon, and Marius Leordeanu. Shift r-cnn: Deep monocular 3d object detection with closed-form geometric constraints. *arXiv preprint arXiv:1905.09970*, 2019. 7
- [38] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *CVPR*, pages 918–927, 2018. 3
- [39] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, pages 652–660, 2017. 3
- [40] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, pages 5099–5108, 2017. 3
- [41] Zengyi Qin, Jinglu Wang, and Yan Lu. Monogrnet: A geometric reasoning network for monocular 3d object localization. In *AAAI*, volume 33, pages 8851–8858, 2019. 1, 2, 7
- [42] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, pages 779–788, 2016. 5
- [43] Thomas Roddick, Alex Kendall, and Roberto Cipolla. Orthographic feature transform for monocular 3d object detection. *arXiv preprint arXiv:1811.08188*, 2018. 3, 7
- [44] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *CVPR*, pages 770–779, 2019. 3
- [45] Andrea Simonelli, Samuel Rota Rota Bulò, Lorenzo Porzi, Manuel López-Antequera, and Peter Kotschieder. Disentangling monocular 3d object detection. *arXiv preprint arXiv:1905.12365*, 2019. 6, 7
- [46] Jie Tang, Fei-Peng Tian, Wei Feng, Jian Li, and Ping Tan. Learning guided convolutional network for depth completion. *arXiv preprint arXiv:1908.01238*, 2019. 3
- [47] Bei Wang, Jianping An, and Jiayan Cao. Voxel-fpn: multi-scale voxel feature aggregation in 3d object detection from point clouds. *arXiv preprint arXiv:1907.05286*, 2019. 3
- [48] Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In *CVPR*, pages 8445–8453, 2019. 1, 3, 7, 8
- [49] Zhixin Wang and Kui Jia. Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection. *arXiv preprint arXiv:1903.01864*, 2019. 3
- [50] Xinshuo Weng and Kris Kitani. Monocular 3d object detection with pseudo-lidar point cloud. *arXiv preprint arXiv:1903.09847*, 2019. 1, 3, 7, 8
- [51] Bichen Wu, Alvin Wan, Xiangyu Yue, Peter Jin, Sicheng Zhao, Noah Golmant, Amir Gholaminejad, Joseph Gonzalez, and Kurt Keutzer. Shift: A zero flop, zero parameter alternative to spatial convolutions. In *CVPR*, pages 9127–9135, 2018. 4
- [52] Jialin Wu, Dai Li, Yu Yang, Chandrajit Bajaj, and Xiangyang Ji. Dynamic filtering with large sampling field for convnets. In *ECCV*, pages 185–200, 2018. 3
- [53] Yu Xiang, Wongun Choi, Yuanqing Lin, and Silvio Savarese. Data-driven 3d voxel patterns for object category recognition. In *CVPR*, pages 1903–1911, 2015. 3, 6
- [54] Bin Xu and Zhenzhong Chen. Multi-level fusion based 3d object detection from monocular images. In *CVPR*, pages 2345–2353, 2018. 2
- [55] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. 3
- [56] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015. 5
- [57] Muhammad Zeeshan Zia, Michael Stark, and Konrad Schindler. Are cars just 3d boxes?:jointly estimating the 3d shape of multiple objects. In *CVPR*, pages 3678–3685, 2014. 3
- [58] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, pages 6848–6856, 2018. 4
- [59] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *CVPR*, pages 4490–4499, 2018. 3