Categorical Depth Distribution Network for Monocular 3D Object Detection

Cody Reading Ali Harakeh Julia Chae Steven L. Waslander
University of Toronto Robotics Institute
{cody.reading, nayoung.chae}@mail.utoronto.ca, ali.harakeh@utoronto.ca, stevenw@utias.utoronto.ca

Abstract

Monocular 3D object detection is a key problem for autonomous vehicles, as it provides a solution with simple configuration compared to typical multi-sensor systems. The main challenge in monocular 3D detection lies in accurately predicting object depth, which must be inferred from object and scene cues due to the lack of direct range measurement. Many methods attempt to directly estimate depth to assist in 3D detection, but show limited performance as a result of depth inaccuracy. Our proposed solution, Categorical Depth Distribution Network (CaDDN), uses a predicted categorical depth distribution for each pixel to project rich contextual feature information to the appropriate depth interval in 3D space. We then use the computationally efficient bird's-eye-view projection and single-stage detector to produce the final output detections. We design CaDDN as a fully differentiable end-to-end approach for joint depth estimation and object detection. We validate our approach on the KITTI 3D object detection benchmark, where we rank 1st among published monocular methods. We also provide the first monocular 3D detection results on the newly released Waymo Open Dataset. We provide a code release for CaDDN which is made available.

1. Introduction

Perception in 3D space is a key component in fields such as autonomous vehicles and robotics, enabling systems to understand their environment and react accordingly. LiDAR [21, 50, 51] and stereo [46, 45, 28, 11] sensors have a long history of use for 3D perception tasks, showing excellent results on 3D object detection benchmarks such as the KITTI 3D object detection benchmark [16] due to their ability to generate precise 3D measurements.

Monocular based 3D perception has been pursued simultaneously, motivated by the potential for a low-cost, easy-to-deploy solution with a single camera [9, 40, 5, 22]. Performance on the same 3D object detection benchmarks lags significantly relative to LiDAR and stereo methods, due to the loss of depth information when scene information is projected onto the image plane.

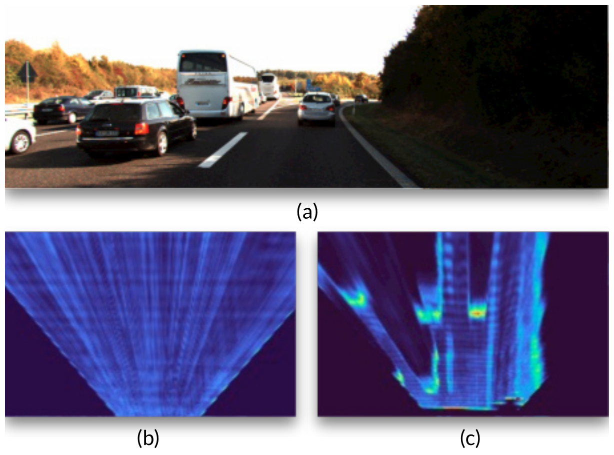

Figure 1. (a) Input image. (b) Without depth distribution supervision, BEV features from CaDDN suffer from smearing effects. (c) Depth distribution supervision encourages BEV features from CaDDN to encode meaningful depth confidence, in which objects can be accurately detected.

To combat this effect, monocular object detection methods [13, 36, 37, 59] often learn depth explicitly, by training a monocular depth estimation network in a separate stage. However, depth estimates are consumed directly in the 3D object detection stage without an understanding of depth confidence, leading to networks that tend to be over-confident in depth predictions. Over-confidence in depth is particularly an issue at long range [59], leading to poor localization. Further, depth estimation is separated from 3D detection during the training phase, preventing depth map estimates from being optimized for the detection task.

Depth information in image data can also be learned implicitly, by directly transforming features from images to 3D space and finally to bird's-eye-view (BEV) grids [48, 44]. Implicit methods, however, tend to suffer from feature smearing, wherein similar image features can exist at multiple locations in the projected space. Feature smearing increases the difficulty of localizing objects in the scene.

To resolve the identified issues, we propose a monocular 3D object detection method, CaDDN, that enables accurate 3D detection by learning categorical depth distributions. By leveraging probabilistic depth estimation, CaDDN is able to generate high quality bird's-eye-view feature representations from images in an end-to-end fashion. We summarize our approach with three contributions.

(1) Categorical Depth Distributions. In order to perform 3D detection, we predict pixel-wise categorical depth distributions to accurately locate image information in 3D space. Each predicted distribution describes the probabilities that a pixel belongs to a set of predefined depth bins. We encourage our distributions to be as sharp as possible around the correct depth bins, in order to encourage our network to focus more on image information where depth estimation is both accurate and confident [23]. By doing so, our network is able to produce sharper and more accurate features that are useful for 3D detection (see Figure 1). On the other hand, our network retains the ability to produce less sharp distributions when depth estimation confidence is low. Using categorical distributions allows our feature encoding to capture the inherent depth estimation uncertainty to reduce the impact of erroneous depth estimates, a property shown to be key to CaDDN's improved performance in Section 4.3. Sharpness in our predicted depth distributions is encouraged through supervision with one-hot encodings of the correct depth bin, which can be generated by projecting LiDAR depth data into the camera frame.

(2) End-To-End Depth Reasoning. We learn depth distributions in an end-to-end fashion, jointly optimizing for accurate depth prediction as well as accurate 3D object detection. We argue that joint depth estimation and 3D detection reasoning encourages depth estimates to be optimized for the 3D detection task, leading to increased performance as shown in Section 4.3.

(3) BEV Scene Representation. We introduce a novel method to generate high quality bird's-eye-view scene representations from single images using categorical depth distributions and projective geometry. We select the bird's-eye-view representation due to its ability to produce excellent 3D detection performance with high computational efficiency [26]. The generated bird's-eye-view representation is used as input to a bird's-eye-view based detector to produce the final output.

CaDDN is shown to rank first among all previously published monocular methods on the Car and Pedestrian categories of the KITTI 3D object detection test benchmark [1], with margins of 1.69% and 1.46% $\text{AP}|_{R_{40}}$ respectively. We are the first to report monocular 3D object detection results on the Waymo Open Dataset [56].

2. Related Work

Monocular Depth Estimation. Monocular depth estimation is performed by generating a single depth value for every pixel in an image. As such, many monocular depth estimation methods are based on architectures used in well-studied pixel-to-pixel mapping problems such as semantic segmentation. As an example, fully convolutional networks (FCNs) [34] were introduced for semantic segmentation, and were subsequently adopted for monocular depth estimation [25]. The atrous spatial pyramid pooling (ASPP) module was also first proposed for semantic segmentation in DeepLab [8, 7, 6] and subsequently used for depth estimation in DORN [15] and BTS [27]. Further, many methods jointly perform depth estimation and segmentation [63, 66, 58, 14] in an end-to-end manner. We follow the design of the semantic segmentation network DeepLabV3 [6] for estimating categorical depth distributions for each pixel in the image.

BEV Semantic Segmentation. BEV segmentation methods [42, 49] attempt to predict BEV semantic maps of 3D scenes from images. Images can be used to either directly estimate BEV semantic maps [39, 35, 60] or to estimate a BEV feature representation [44, 47, 41] as an intermediate step for the segmentation task. In particular, Lift, Splat, Shoot [44] predicts categorical depth distributions in an unsupervised manner, in order to generate intermediate BEV representations. In this work, we predict categorical depth distributions via supervision with ground truth one-hot encodings to generate more accurate depth distributions for object detection.

Monocular 3D Detection. Monocular 3D object detection methods often generate intermediate representations to assist in the 3D detection task. Based on these representations, monocular detection can be divided into three categories: direct, depth-based, and grid-based methods.

Direct Methods. Direct methods [9, 52, 4, 32] estimate 3D detections directly from images without predicting an intermediate 3D scene representation. Rather, direct methods [53, 12, 40, 3] can incorporate the geometric relationship between the 2D image plane and 3D space to assist with detections. For example, object keypoints can be estimated on the image plane, in order to assist in 3D box construction using known geometry [33, 29]. M3D-RPN [3] introduces depth-aware convolutions that divides the input row-wise and learns non-shared kernels for each region, to learn location specific features that correlate to regions in 3D space. Shape estimation can be performed for objects in the scene to create an understanding of 3D object geometry. Shape estimates can be supervised from labeled vertices of 3D CAD models [5, 24], from LiDAR scans [22], or directly from input data in a self-supervised manner [2]. A drawback for direct methods is that detections are generated directly from 2D images, without access to explicit depth information, usually resulting in reduced performance in localization relative to other methods.

Depth-Based Methods. Depth-based methods perform the

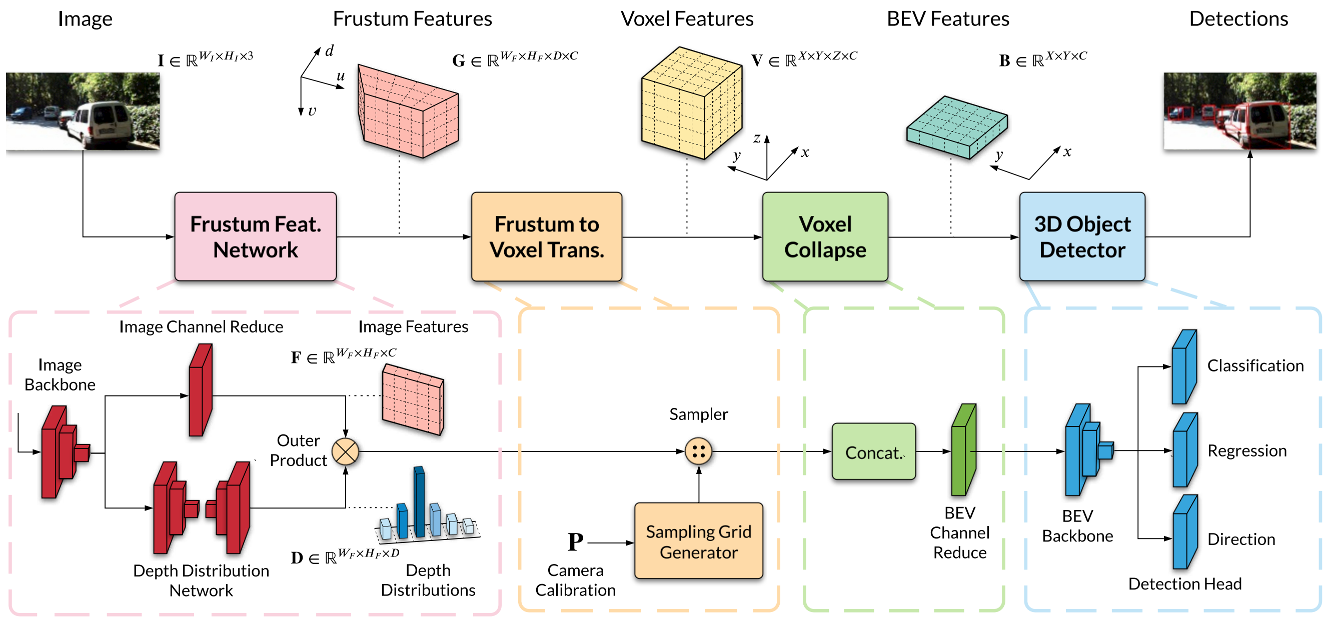

Image
Frustum Features
Voxel Features
BEV Features
Detections
$\mathbf{I} \in \mathbb{R}^{W_I \times H_I \times 3}$
$\mathbf{G} \in \mathbb{R}^{W_F \times H_F \times D \times C}$
$\mathbf{V} \in \mathbb{R}^{X \times Y \times Z \times C}$
$\mathbf{B} \in \mathbb{R}^{X \times Y \times C}$
Frustum Feat. Network
Frustum to Voxel Trans.
Voxel Collapse
3D Object Detector
Image Channel Reduce
Image Features
Image Backbone
$\mathbf{F} \in \mathbb{R}^{W_F \times H_F \times C}$
Outer Product
Depth Distribution Network
$\mathbf{D} \in \mathbb{R}^{W_F \times H_F \times D}$
Depth Distributions
Sampler
P
Camera Calibration
Sampling Grid Generator
Concat.
BEV Channel Reduce
BEV Backbone
Classification
Regression
Direction
Detection Head

Figure 2. CaDDN Architecture. The network is composed of three modules to generate 3D feature representations and one to perform 3D detection. Frustum features $\mathbf{G}$ are generated from an image $\mathbf{I}$ using estimated depth distributions $\mathbf{D}$, which are transformed into voxel features $\mathbf{V}$. The voxel features are collapsed to bird's-eye-view features $\mathbf{B}$ to be used for 3D object detection.

3D detection task using pixel-wise depth maps as an additional input, where the depth maps are precomputed using monocular depth estimation architectures [15]. Estimated depth maps can be used in combination with images to perform the 3D detection task [38, 64, 36, 13]. Alternatively, depth maps can be converted to 3D point clouds, commonly known as Pseudo-LiDAR [59], which are either used directly [61, 65] or combined with image information [62, 37] to generate 3D object detection results. Depth-based methods separate depth estimation from 3D object detection during the training stage, leading to the learning of sub-optimal depth maps used for the 3D detection task. Accurate depth should be prioritized for pixels belonging to objects of interest, and is less important for background pixels, a property that is not captured if depth estimation and object detection are trained independently.

Grid-Based Methods. Grid-based methods avoid estimating raw depth values by predicting a BEV grid [48, 55] representation, to be used as input for 3D detection architectures. Specifically, OFT [48] populates a voxel grid by projecting voxels into the image plane and sampling image features, to be transformed into a BEV representation. Multiple voxels can be projected to the same image feature, leading to repeated features along the projection ray and reduced detection accuracy.

CaDDN addresses all identified issues by jointly performing depth estimation and 3D object detection in an end-to-end manner, and leverages the depth estimates to generate meaningful bird's-eye-view representations with accurate and localized features.

3. Methodology

CaDDN learns to generate BEV representations from images by projecting image features into 3D space. 3D object detection is then performed with the rich BEV representation using an efficient BEV detection network. An overview of CaDDN's architecture is shown in Figure 2.

3.1. 3D Representation Learning

Our network learns to produce BEV representations that are well-suited for the task of 3D object detection. Taking an image as input, we construct a frustum feature grid using the estimated categorical depth distributions. The frustum feature grid is transformed into a voxel grid using known camera calibration parameters, and then collapsed to a bird's-eye-view feature grid.

Frustum Feature Network. The purpose of the frustum feature network is to project image information into 3D space, by associating image features to estimated depths. Specifically, the input to the frustum feature network is an image $\mathbf{I} \in \mathbb{R}^{W_I \times H_I \times 3}$, where W_I, H_I are the width and height of the image. The output is a frustum feature grid $\mathbf{G} \in \mathbb{R}^{W_F \times H_F \times D \times C}$, where W_F, H_F, are the width and height of the image feature representation, D is the number of discretized depth bins, and C is the number of feature channels. We note that the structure of the frustum grid is similar to the plane-sweep volume used in the stereo 3D de-

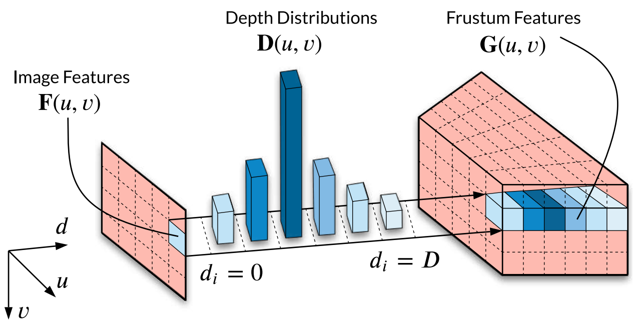

Figure 3. Each feature pixel $\mathbf{F}(u, v)$ is weighted by its depth distribution probabilities $\mathbf{D}(u, v)$ of belonging to D discrete depth bins to generate frustum features $\mathbf{G}(u, v)$.

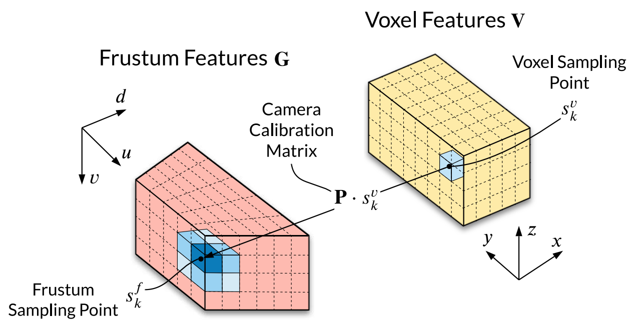

Figure 4. Sampling points in each voxel are projected into the frustum grid. Frustum features are sampled using trilinear interpolation (shown as blue in $\mathbf{G}$) to populate voxels in $\mathbf{V}$.

tection method DSGN [11].

A ResNet-101 [17] backbone is used to extract image features $\tilde{\mathbf{F}} \in \mathbb{R}^{W_F \times H_F \times C}$ (see Image Backbone in Figure 2). In our implementation, we extract the image features from *Block1* of the ResNet-101 backbone in order to maintain a high spatial resolution. A high spatial resolution is necessary for an effective frustum to voxel grid transformation, such that the frustum grid can be finely sampled without repeated features.

The image features $\tilde{\mathbf{F}}$ are used to estimate pixel-wise categorical depth distributions $\mathbf{D} \in \mathbb{R}^{W_F \times H_F \times D}$, where the categories are the D discretized depth bins. Specifically, we predict D probabilities for each pixel in the image features $\tilde{\mathbf{F}}$, where each probability indicates the network's confidence that depth value belongs to a specified depth bin. The definition of the depth bins relies on the depth discretization method as discussed in Section 3.3.

We follow the design of the semantic segmentation network DeepLabV3 [6] to estimate the categorical depth distributions from image features $\tilde{\mathbf{F}}$ (Depth Distribution Network in Figure 2), where we modify the network to produce pixel-wise probability scores of belonging to depth bins rather than semantic classes with a downsample-upsample architecture. Image features $\tilde{\mathbf{F}}$ are downsampled with the remaining components of the ResNet-101 [17] backbone (*Block2*, *Block3*, and *Block4*). An atrous spatial pyramid pooling [6] (ASPP) module is applied to capture multi-scale information, where the number of output channels is set as D. The output of the ASPP module is upsampled to the original feature size with bilinear interpolation to produce the categorical depth distributions $\mathbf{D} \in \mathbb{R}^{W_F \times H_F \times D}$. A softmax function is applied for each pixel to normalize the D logits into probabilities between 0 and 1.

In parallel to estimating depth distributions, we perform channel reduction (Image Channel Reduce in Figure 2) on image features $\tilde{\mathbf{F}}$ to generate the final image features $\mathbf{F}$, using a 1x1 convolution + BatchNorm + ReLU layer to reduce the number of channels from $C = 256$ to $C = 64$. Channel reduction is required to reduce the high memory footprint of ResNet-101 features that will be populated in the 3D frustum grid.

Let (u, v, c) represent a coordinate in image features $\mathbf{F}$ and (u, v, d_i) represent a coordinate in categorical depth distributions $\mathbf{D}$, where (u, v) are the feature pixel location, c is the channel index, and d_i is the depth bin index. To generate a frustum feature grid $\mathbf{G}$, each feature pixel $\mathbf{F}(u, v)$ is weighted by its associated depth bin probabilities in $\mathbf{D}(u, v)$ to populate the depth axis d_i, visualized in Figure 3. Feature pixels can be weighted by depth probability using the outer product, defined as:

$$\mathbf{G}(u, v) = \mathbf{D}(u, v) \otimes \mathbf{F}(u, v) \tag{1}$$

where $\mathbf{D}(u, v)$ is the predicted depth distribution and $\mathbf{G}(u, v)$ is an output matrix of size $D \times C$. The outer product in Equation 1 is computed for each pixel to form frustum features $\mathbf{G} \in \mathbb{R}^{W_F \times H_F \times D \times C}$.

Frustum to Voxel Transformation. The frustum features $\mathbf{G} \in \mathbb{R}^{W_F \times H_F \times D \times C}$ are transformed to a voxel representation $\mathbf{V} \in \mathbb{R}^{X \times Y \times Z \times C}$ leveraging known camera calibration and differentiable sampling, shown in Figure 4. Voxel sampling points $s_k^v = [x, y, z]_k^T$ are generated at the center of each voxel and transformed to the frustum grid to form frustum sampling points $\tilde{s}_k^f = [u, v, d_c]_k^T$, where d_c is the continuous depth value along the frustum depth axis d_i. The transformation is performed using the camera calibration matrix $\mathbf{P} \in \mathbb{R}^{3 \times 4}$. Each continuous depth value d_c is converted to a discrete depth bin index d_i using the depth discretization method outlined in Section 3.3. Frustum features in $\mathbf{G}$ are sampled using sampling points $s_k^f = [u, v, d_i]_k^T$ with trilinear interpolation (shown in blue in Figure 4) to populate voxel features in $\mathbf{V}$.

The spatial resolution of the frustum grid $\mathbf{G}$ and the voxel grid $\mathbf{V}$ should be similar for an effective transformation. A high resolution voxel grid $\mathbf{V}$ leads to a high density of sampling points that will oversample a low resolution frustum grid, resulting in a large amount of similar voxel features. Therefore, we extract the features $\tilde{\mathbf{F}}$ from *Block1* of the ResNet-101 backbone to ensure our frustum grid $\mathbf{G}$ is of high spatial resolution.

Voxel Collapse to BEV. The voxel features $\mathbf{V} \in \mathbb{R}^{X \times Y \times Z \times C}$ are collapsed to a single height plane to generate bird's-eye-view features $\mathbf{B} \in \mathbb{R}^{X \times Y \times C}$. BEV grids greatly reduce the computational overhead while offering similar detection performance to 3D voxel grids [26], motivating their use in our network. We concatenate the vertical axis z of the voxel grid $\mathbf{V}$ along the channel dimension c to form a BEV grid $\tilde{\mathbf{B}} \in \mathbb{R}^{X \times Y \times Z * C}$. The number of channels are reduced using a 1x1 convolution + BatchNorm + ReLU layer (see BEV Channel Reduce in Figure 2), which retrieves the original number of channels C while learning the relative importance of each height slice, resulting in a BEV grid $\mathbf{B} \in \mathbb{R}^{X \times Y \times C}$.

3.2. BEV 3D Object Detection

To perform 3D object detection on the BEV feature grid, we adopt the backbone and detection head of the well-established BEV 3D object detector PointPillars [26], as it has been shown to provide accurate 3D detection results with a low computational overhead. For the BEV backbone, we increase the number of 3x3 convolution + BatchNorm + ReLU layers in the downsample blocks from (4, 6, 6) used in the original PointPillars [26] to (10, 10, 10) for *Block1*, *Block2*, and *Block3* respectively. Increasing the number of convolutional layers expands the learning capacity in our BEV network, important for learning from lower quality features produced by images compared to higher quality features originally produced by LiDAR point clouds. We use the same detection head as PointPillars [26] to generate our final detections.

3.3. Depth Discretization

The continuous depth space is discretized in order to define the set of D bins used in the depth distributions $\mathbf{D}$. Depth discretization can be performed with uniform discretization (UD) with a fixed bin size, spacing-increasing discretization (SID) [15] with increasing bin sizes in log space, or linear-increasing discretization (LID) [57] with linearly increasing bin sizes. Depth discretization techniques are visualized in Figure 5. We adopt LID as our depth discretization as it provides balanced depth estimation for all depths [57]. LID is defined as:

$$d_c = d_{\min} + \frac{d_{\max} - d_{\min}}{D(D+1)} \cdot d_i(d_i + 1) \tag{2}$$

where d_c is the continuous depth value, $[d_{\min}, d_{\max}]$ is the full depth range to be discretized, D is the number of depth bins, and d_i is the depth bin index.

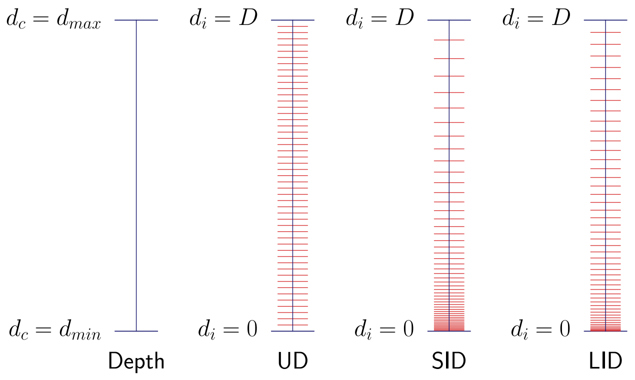

Figure 5. Depth Discretization Methods. Depth d_c is discretized over a depth range $[d_{min}, d_{max}]$ into D discrete bins. Commonly used methods include uniform (UD), spacing-increasing (SID), and linear-increasing (LID) discretization.

3.4. Depth Distribution Label Generation

We require depth distribution labels $\hat{\mathbf{D}}$ in order to supervise our predicted depth distributions. Depth distribution labels are generated by projecting LiDAR point clouds into the image frame to create sparse dense maps. Depth completion [20] is performed to generate depth values at each pixel in the image. We require depth information at each image feature pixel, so we downsample the depth maps of size $W_I \times H_I$ to the image feature size $W_F \times H_F$. The depth maps are converted to bin indices using the LID discretization method described in Section 3.3, followed by a conversion into a one-hot encoding to generate the depth distribution labels. A one-hot encoding ensures the depth distribution labels are sharp, essential to encourage sharpness in our depth distribution predictions via supervision.

3.5. Training Losses

Generally, classification is performed by predicting categorical distributions, and encouraging sharpness in the distribution in order to select the correct class [19]. We leverage classification to encourage a single correct depth bin when supervising the depth distribution network, using the focal loss [30]:

$$L_{\text{depth}} = \frac{1}{W_F \cdot H_F} \sum_{u=1}^{W_F} \sum_{v=1}^{H_F} \text{FL}(\mathbf{D}(u,v), \hat{\mathbf{D}}(u,v)) \tag{3}$$

where $\mathbf{D}$ is the depth distribution predictions and $\hat{\mathbf{D}}$ is the depth distribution labels. We found that autonomous driving datasets contain images with fewer object pixels than background pixels, leading to loss functions that prioritize background pixels when all pixel losses are weighted evenly. We set the focal loss [30] weighting factor α as $\alpha_{\text{fg}} = 3.25$ for foreground object pixels and $\alpha_{\text{bg}} = 0.25$ for background pixels. Foreground object pixels are deter-

Method	Frames	Car (IOU = 0.7)			Pedestrian (IOU = 0.5)			Cyclist (IOU = 0.5)		
		Easy	**Mod.**	Hard	Easy	**Mod.**	Hard	Easy	**Mod.**	Hard
Kinematic3D [4]	4	19.07	12.72	9.17	–	–	–	–	–	–
OFT [48]	1	1.61	1.32	1.00	0.63	0.36	0.35	0.14	0.06	0.07
ROI-10D [38]	1	4.32	2.02	1.46	–	–	–	–	–	–
MonoPSR [22]	1	10.76	7.25	5.85	6.12	4.00	3.30	**8.37**	**4.74**	**3.68**
Mono3D-PLiDAR [61]	1	10.76	7.50	6.10	–	–	–	–	–	–
MonoDIS [52]	1	10.37	7.94	6.40	–	–	–	–	–	–
UR3D [64]	1	15.58	8.61	6.00	–	–	–	–	–	–
M3D-RPN [3]	1	14.76	9.71	7.42	4.92	3.48	2.94	0.94	0.65	0.47
SMOKE [33]	1	14.03	9.76	7.84	–	–	–	–	–	–
MonoPair [12]	1	13.04	9.99	8.65	**10.02**	**6.68**	**5.53**	3.79	2.12	1.83
RTM3D [29]	1	14.41	10.34	8.77	–	–	–	–	–	–
AM3D [37]	1	16.50	10.74	9.52	–	–	–	–	–	–
MoVi-3D [53]	1	15.19	10.90	9.26	8.99	5.44	4.57	1.08	0.63	0.70
RAR-Net [32]	1	16.37	11.01	9.52	–	–	–	–	–	–
PatchNet [36]	1	15.68	11.12	**10.17**	–	–	–	–	–	–
DA-3Ddet [65]	1	**16.77**	11.50	8.93	–	–	–	–	–	–
D4LCN [13]	1	16.65	**11.72**	9.51	4.55	3.42	2.83	2.45	1.67	1.36
CaDDN (ours)	1	**19.17**	**13.41**	**11.46**	**12.87**	**8.14**	**6.76**	**7.00**	**3.41**	**3.30**
Improvement	–	*+2.40*	*+1.69*	*+1.29*	*+2.85*	*+1.46*	*+1.23*	*-1.37*	*-1.33*	*-0.38*

Table 1. 3D detection results on the KITTI [16] *test* set. Results are shown using the $\text{AP}|_{R_{40}}$ metric only for results that are readily available. We indicate the highest result with **red** and the second highest with **blue**.

mined as all pixels that lie within 2D object bounding box labels, and background pixels are all remaining pixels. We set the focal loss [30] focusing parameter $\gamma = 2.0$.

We use the classification loss L_{cls}, regression loss L_{reg}, and direction classification loss L_{dir} from PointPillars [26] for 3D object detection. The total loss of our network is the combination of the depth and 3D detection losses:

$$L = \lambda_{\text{depth}} L_{\text{depth}} + \lambda_{\text{cls}} L_{\text{cls}} + \lambda_{\text{reg}} L_{\text{reg}} + \lambda_{\text{dir}} L_{\text{dir}} \quad (4)$$

where $\lambda_{\text{depth}}, \lambda_{\text{cls}}, \lambda_{\text{reg}}, \lambda_{\text{dir}}$ are fixed loss weighting factors.

4. Experimental Results

To demonstrate the effectiveness of CaDDN we present results on both the KITTI 3D object detection benchmark [16] and the Waymo Open Dataset [56].

The KITTI 3D object detection benchmark [16] is divided into 7,481 training samples and 7,518 testing samples. The training samples are commonly divided into a *train* set (3,712 samples) and a *val* set (3,769 samples) following [10], which is also adopted here. We compare CaDDN with existing methods on the *test* set by training our model on both the *train* and *val* sets. We evaluate on the *val* set for ablation by training our model on only the *train* set.

The Waymo Open Dataset [56] is a more recently released autonomous driving dataset, which consists of 798 training sequences and 202 validation sequences. The dataset also includes 150 test sequences without ground truth data. The dataset provides object labels in the full 360° field of view with a multi-camera rig. We only use the front camera and only consider object labels in the front-camera's field of view (50.4°) for the task of monocular object detection, and provide results on the validation sequences. We sample every 3rd frame from the training sequences to form our training set (51,564 samples) due to the large dataset size and high frame rate.

Input Parameters. The voxel grid is defined by a range and voxel size in 3D space. On KITTI [16], we use $[2, 46.8] \times [-30.08, 30.08] \times [-3, 1]$ (m) for the range and $[0.16, 0.16, 0.16]$ (m) for the voxel size for the x, y, and z axes respectively. On Waymo, we use $[2, 55.76] \times [-25.6, 25.6] \times [-4, 4]$ (m) for the range and $[0.16, 0.16, 0.16]$ (m) for the voxel size. Additionally, we downsample Waymo images to 1248×832.

Training and Inference Details. Our method is implemented in PyTorch [43]. The network is trained on a NVIDIA Tesla V100 (32G) GPU. The Adam [18] optimizer is used with an initial learning rate of 0.001 and is modified using the one-cycle learning rate policy [54]. We train the model for 80 epochs on the KITTI dataset [16] and 10 epochs on the Waymo Open Dataset [56]. We use a batch size of 4 for KITTI [16] and a batch size of 2 for Waymo. The values $\lambda_{\text{depth}} = 3.0, \lambda_{\text{cls}} = 1.0, \lambda_{\text{reg}} = 2.0, \lambda_{\text{dir}} = 0.2$ are used for the loss weighting factors in Equation 4. We employ horizontal flip as our data augmentation and train one model for all classes. During inference, we filter boxes

Difficulty	Method	3D mAP				3D mAPH			
		Overall	0 - 30m	30 - 50m	50m - ∞	Overall	0 - 30m	30 - 50m	50m - ∞
LEVEL_1 (IOU = 0.7)	M3D-RPN [3]	0.35	1.12	0.18	0.02	0.34	1.10	0.18	0.02
	CaDNN (Ours)	**5.03**	**14.54**	**1.47**	**0.10**	**4.99**	**14.43**	**1.45**	**0.10**
	Improvement	*+4.69*	*+13.43*	*+1.28*	*+0.08*	*+4.65*	*+13.33*	*+1.28*	*+0.08*
LEVEL_2 (IOU = 0.7)	M3D-RPN [3]	0.33	1.12	0.18	0.02	0.33	1.10	0.17	0.02
	CaDNN (Ours)	**4.49**	**14.50**	**1.42**	**0.09**	**4.45**	**14.38**	**1.41**	**0.09**
	Improvement	*+4.15*	*+13.38*	*+1.24*	*+0.07*	*+4.12*	*+13.28*	*+1.24*	*+0.07*
LEVEL_1 (IOU = 0.5)	M3D-RPN [3]	3.79	11.14	2.16	0.26	3.63	10.70	2.09	0.21
	CaDNN (Ours)	**17.54**	**45.00**	**9.24**	**0.64**	**17.31**	**44.46**	**9.11**	**0.62**
	Improvement	*+13.76*	*+33.86*	*+7.08*	*+0.39*	*+13.69*	*+33.77*	*+7.02*	*+0.41*
LEVEL_2 (IOU = 0.5)	M3D-RPN [3]	3.61	11.12	2.12	0.24	3.46	10.67	2.04	0.20
	CaDNN (Ours)	**16.51**	**44.87**	**8.99**	**0.58**	**16.28**	**44.33**	**8.86**	**0.55**
	Improvement	*+12.89*	*+33.75*	*+6.87*	*+0.34*	*+12.82*	*+33.66*	*+6.81*	*+0.36*

Table 2. Results on the Waymo Open Dataset Validation Set on the Vehicle class. We evaluate M3D-RPN [3] as a baseline for comparison.

with a score threshold of 0.1 and apply non-maximum suppression (NMS) with an IoU threshold of 0.01.

4.1. KITTI Dataset Results

Results on the KITTI dataset [16] are evaluated using average precision $(\text{AP}|_{R_{40}})$. The evaluation is separated by difficulty settings (Easy, Moderate, and Hard) and by object class (Car, Pedestrian, and Cyclist). The Car class has an IoU criteria of 0.7 while the Pedestrian and Cyclist classes have an IoU criteria of 0.5, where IoU criteria is a threshold to be considered a true positive detection.

Table 1 shows the results of CaDDN on the KITTI [16] *test* set compared to state-of-the-art published monocular methods, listed in rank order of performance on the Car class at the Moderate difficulty setting. We note that our method outperforms previous single frame methods by large margins on $\text{AP}|_{R_{40}}$ of +2.40%, +1.69%, and +1.29% on the Car class on the Easy, Moderate, and Hard difficulties respectively. Additionally, CaDDN ranks higher than the multi-frame method Kinematic3D [4]. Our method also outperforms the previous state-of-the art method on the Pedestrian class MonoPair [12] with margins on $\text{AP}|_{R_{40}}$ of +2.85%, +1.46%, and +1.23%. Our method achieves second place on the Cyclist class with margins on $\text{AP}|_{R_{40}}$ of -1.37%, -1.33%, and -0.38% relative to MonoPSR [22].

4.2. Waymo Dataset Results

We adopt the officially released evaluation to calculate the mean average precision (mAP) and the mean average precision weighted by heading (mAPH) on the Waymo Open Dataset [56]. The evaluation is separated by difficulty setting (LEVEL_1, LEVEL_2) and distance to the sensor (0 - 30m, 30 - 50m, and 50m - ∞). We evaluate on the Vehicle class with an IoU criteria of 0.7 and 0.5.

To the best of our knowledge, no monocular methods have reported results on Waymo. In order to provide a baseline, we extend the official implementation of M3D-RPN [3] to support the Waymo Open Dataset [56]. Table 2 shows the results of both the M3D-RPN [3] baseline and CaDDN on the Waymo validation set. Our method significantly outperforms M3D-RPN [3] with margins on AP/APH of +4.69%/+4.65% and +4.15%/+4.12% on the LEVEL_1 and LEVEL_2 difficulties respectively for an IoU criteria of 0.7.

Exp.	**D**	L_{depth}	α_{fg}	LID	Car (IOU = 0.7)		
					Easy	Mod.	Hard
1					7.83	5.66	4.84
2	✓				9.33	6.43	5.30
3	✓	✓			19.73	14.03	11.84
4	✓	✓	✓		20.40	15.10	12.75
5	✓	✓	✓	✓	**23.57**	**16.31**	**13.84**

Table 3. CaDDN Ablation Experiments on the KITTI *val* set using $\text{AP}|_{R_{40}}$. **D** indicates depth distribution prediction, L_{depth} indicates depth distribution supervision. α_{fg} indicates separate setting of loss weighting factor for foreground object pixels in the depth loss function L_{depth}. LID indicates the LID discretization method.

Exp.	**D**	L_{depth}	$\otimes$	Car (IOU = 0.7)		
				Easy	Mod.	Hard
1	BTS [27]	Sep.		16.69	10.18	8.63
2	DORN [15]	Sep.		16.43	11.04	9.65
3	CaDDN	Sep.		17.64	12.26	10.10
4	CaDDN	Joint		20.61	13.71	11.96
5	CaDDN	Joint	✓	**23.57**	**16.31**	**13.84**

Table 4. CaDDN Depth Estimation Ablation on the KITTI *val* set using $\text{AP}|_{R_{40}}$. **D** indicates the source of the depth estimates used to generate depth distributions. L_{depth} indicates if depth estimation and object detection are seperately or jointly optimized. $\otimes$ indicates if full distributions are used to generate frustum features **G**.

4.3. Ablation Studies

We provide ablation studies on our network to validate our design choices. The results are shown in Tables 3 and 4.

Sharpness in Depth Distributions. Experiment 1 in Table 3 shows the detection performance when frustum features $\mathbf{G}$ are populated by repeating image features $\mathbf{F}$ along depth axis d_i. Experiment 2 adds depth distribution predictions $\mathbf{D}$ to separately weigh image features $\mathbf{F}$, which improves performance on $\text{AP}|_{R_{40}}$ by +1.50%, +0.77%, and +0.46% on the Car class on the Easy, Moderate, and Hard difficulties respectively. Performance is greatly increased (+10.40%, +7.60%, +6.54%) once depth distribution supervision is added in Experiment 3 validating its inclusion. The addition of depth distribution supervision encourages sharp and accurate categorical depth distributions, that encourages image information to be located in 3D space where depth estimation is both accurate and confident. Encouraging sharpness around correct depth bins results in object features that are uniquely located and easily distinguished (see Figure 1) in the BEV projection.

Object Weighting for Depth Distribution Estimation. Experiments 1, 2, and 3 in Table 3 use a fixed loss weighting factor $\alpha = 0.25$ for all pixels in the depth loss function L_{depth}. Experiment 4 shows an improvement (+0.67%, +1.07%, +0.91%) after depth loss weights $\alpha_{\text{fg}} = 3.25/\alpha_{\text{bg}} = 0.25$ are set seperately for foreground object and background pixels (see Section 3.5). Setting a larger foreground object weighting factor α_{fg} encourages depth estimation to be prioritized for object pixels, leading to more accurate depth estimation and localization for objects.

Linear Increasing Discretization. Experiment 5 in Table 3 shows the detection performance improvement (+3.17%, +1.21%, +1.09%) when LID (see Section 3.3) is used rather than uniform discretization. We attribute the performance increase to the accurate depth estimation LID provides across all depths [57].

Joint Depth Understanding. Experiments 1, 2 and 3 in Table 4 show the detection performance with separate depth estimation from BTS [27], DORN [15], and CaDDN respectively. The depth maps from BTS [27] and DORN [15] are converted to depth bin indices using LID discretization as outlined in Section 3.3, and converted to a one-hot encoding to generate the depth distributions $\mathbf{D}$. The one-hot encoding places the image feature at a single depth bin indicated by the input depth map when generating frustum features $\mathbf{G}$. We constuct an equivalent version of CaDDN that selects a single depth bin for each pixel, by selecting the bin with highest probability for each distribution in $\mathbf{D}$. Experiment 4 shows improved performance (+2.97%, +1.45%, +1.86%) when depth estimation and object detection are performed jointly, which we attribute to the well-known benefits of end-to-end learning for 3D detection.

Categorical Depth Distributions. Experiment 5 in Table 4 uses the full depth distribution $\mathbf{D}$ in the frustum features computation $\mathbf{G} = \mathbf{D} \otimes \mathbf{F}$, leading to a clear increase in performance (+2.96%, 2.60%, 1.88%). We attribute the performance increase to the additional depth uncertainty information embedded in the feature representations.

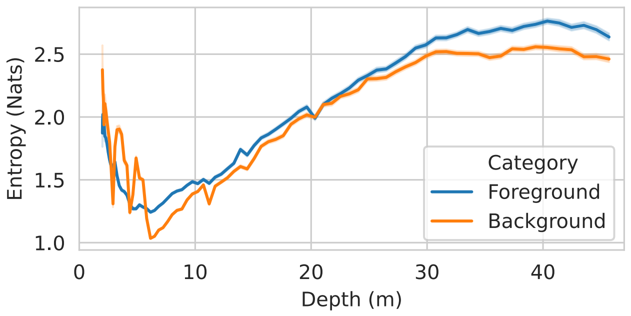

Figure 6. We plot the entropy of the estimated depth distributions $\mathbf{D}$ against depth. We show both the mean (solid line) and 95% confidence interval (shaded region) at each ground truth depth bin.

4.4. Depth Distribution Uncertainty

To validate that our depth distributions contain meaningful uncertainty information, we compute the Shanon entropy for each estimated categorical depth distribution in $\mathbf{D}$. We label each distribution with its associated ground truth depth bin and foreground/background classification. For each group, we compute the entropy statistics which are shown in Figure 6. We observe that entropy generally increases as a function of depth, where depth estimates are challenging, indicating our distributions describe meaningful uncertainty information. Our network produces the lowest distribution entropy at pixels with ground truth depth of around 6 meters. We attribute the high entropy at depths closer than 6 meters to the small number of pixels at shorter ranges in the training set. Finally, we note that the foreground depth distribution estimates have slightly higher entropy than background pixels, a phenomenon that can also be attributed to training set imbalance.

5. Conclusion

We have presented CaDDN, a novel monocular 3D object detection method that estimates accurate categorical depth distributions for each pixel. The depth distributions are combined with the image features to generate bird's-eye-view representations that retain depth confidence, to be exploited for 3D object detection. We have shown that estimating sharp categorical distributions centered around the correct depth value, and jointly performing depth estimation and object detection is vital for 3D object detection performance, leading to a 1st place ranking on the KITTI dataset [1] among all published methods at the time of submission.

References

[1] Kitti's 3d object detection evaluation benchmark 2017. http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d. Accessed on 15.11.2020. 2, 8

[2] Deniz Beker, Hiroharu Kato, Mihai Adrian Morariu, Takahiro Ando, Toru Matsuoka, Wadim Kehl, and Adrien Gaidon. Monocular differentiable rendering for self-supervised 3d object detection. *ECCV*, 2020. 2

[3] Garrick Brazil and Xiaoming Liu. M3D-RPN: monocular 3D region proposal network for object detection. *ICCV*, 2019. 2, 6, 7, 11

[4] Garrick Brazil, Gerard Pons-Moll, Xiaoming Liu, and Bernt Schiele. Kinematic 3d object detection in monocular video. *ECCV*, 2020. 2, 6, 7, 11

[5] 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. *CVPR*, 2017. 1, 2

[6] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint*, 2017. 2, 4, 11

[7] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *ICLR*, 2015. 2

[8] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint*, 2016. 2

[9] Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma1, Sanja Fidler, and Raquel Urtasun. Monocular 3d object detection for autonomous driving. *CVPR*, 2016. 1, 2

[10] Xiaozhi Chen, Kaustav Kundu, Yukun Zhu, Andrew G Berneshawi, Huimin Ma, Sanja Fidler, and Raquel Urtasun. 3d object proposals for accurate object class detection. *NIPS*, 2015. 6

[11] Yilun Chen, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Dsgn: Deep stereo geometry network for 3d object detection. *CVPR*, 2020. 1, 4

[12] Yongjian Chen, Lei Tai, Kai Sun, and Mingyang Li. Monopair: Monocular 3d object detection using pairwise spatial relationships. *CVPR*, 2020. 2, 6, 7, 11

[13] Mingyu Ding, Yuqi Huo, Hongwei Yi, Zhe Wang, Jianping Shi, Zhiwu Lu, and Ping Luo. Learning depth-guided convolutions for monocular 3d object detection. *CVPR*, 2020. 1, 3, 6, 11

[14] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. *ICCV*, 2015. 2

[15] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. *CVPR*, 2018. 2, 3, 5, 7, 8

[16] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. *CVPR*, 2012. 1, 6, 7, 11

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CVPR*, 2016. 4

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015. 6

[19] S. B. Kotsiantis. Supervised machine learning: A review of classification techniques. In *Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in EHealth, HCI, Information Retrieval and Pervasive Technologies*, NLD, 2007. IOS Press. 5

[20] Jason Ku, Ali Harakeh, and Steven Lake Waslander. In defense of classical image processing: Fast depth completion on the CPU. *CRV*, 2018. 5

[21] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven Lake Waslander. Joint 3D proposal generation and object detection from view aggregation. *IROS*, 2018. 1

[22] Jason Ku, Alex D. Pon, and Steven L. Waslander. Monocular 3D object detection leveraging accurate proposals and shape reconstruction. *CVPR*, 2019. 1, 2, 6, 7, 11

[23] Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. Accurate uncertainties for deep learning using calibrated regression. *arXiv preprint arXiv:1807.00263*, 2018. 2

[24] Abhijit Kundu, Yin Li, and James M. Rehg. 3D-RCNN: Instance-level 3D object reconstruction via render-and-compare. *CVPR*, 2018. 2

[25] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. *3DV*, 2016. 2

[26] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. PointPillars: Fast encoders for object detection from point clouds. *CVPR*, 2019. 2, 5, 6

[27] Jin Han Lee, Myung-Kyu Han, Dong Wook Ko, and Il Hong Suh. From big to small: Multi-scale local planar guidance for monocular depth estimation. *arXiv preprint*, 2019. 2, 7, 8

[28] Chengyao Li, Jason Ku, and Steven L. Waslander. Confidence guided stereo 3d object detection with split depth estimation. *IROS*, 2020. 1

[29] Peixuan Li, Huaici Zhao, Pengfei Liu, and Feidao Cao. Rtm3d: Real-time monocular 3d detection from object keypoints for autonomous driving. *ECCV*, 2020. 2, 6, 11

[30] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *PAMI*, 2018. 5, 6

[31] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *ECCV*, 2014. 11

[32] Lijie Liu, Chufan Wu, Jiwen Lu, Lingxi Xie, Jie Zhou, and Qi Tian. Reinforced axial refinement network for monocular 3d object detection. *ECCV*, 2020. 2, 6, 11

[33] Zechen Liu, Zizhang Wu, and Roland Toth. Smoke: Single-stage monocular 3d object detection via keypoint estimation. *CVPRW*, 2020. 2, 6, 11

[34] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CVPR*, 2015. 2

[35] Chenyang Lu, Gijs Dubbelman, and Marinus Jacobus Gerardus van de Molengraft. Monocular semantic occupancy grid mapping with convolutional variational auto-encoders. *ICRA*, 2019. 2

[36] Xinzhu Ma, Shinan Liu, Zhiyi Xia, Hongwen Zhang, Xingyu Zeng, and Wanli Ouyang. Rethinking pseudo-lidar representation. *ECCV*, 2020. 1, 3, 6, 11

[37] Xinzhu Ma, Zhihui Wang, Haojie Li, Wanli Ouyang, and Pengbo Zhang. Accurate monocular 3D object detection via color-embedded 3D reconstruction for autonomous driving. *ICCV*, 2019. 1, 3, 6, 11

[38] Fabian Manhardt, Wadim Kehl, and Adrien Gaidon. ROI-10D: monocular lifting of 2d detection to 6d pose and metric shape. *CVPR*, 2019. 3, 6, 11

[39] Kaustubh Mani, Swapnil Daga, Shubhika Garg, Sai Shankar Narasimhan, Madhava Krishna, and Krishna Murthy Jatavallabhula. Monolayout: Amodal scene layout from a single image. *WACV*, 2020. 2

[40] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka. 3d bounding box estimation using deep learning and geometry. *CVPR*, 2016. 1, 2

[41] Mong H. Ng, Kaahan Radia, Jianfei Chen, Dequan Wang, Ionel Gog, and Joseph E. Gonzalez. Bev-seg: Bird's eye view semantic segmentation using geometry and semantic point cloud. *CVPRW*, 2020. 2

[42] Bowen Pan, Jiankai Sun, Alex Andonian, Aude Oliva, and Bolei Zhou. Cross-view semantic segmentation for sensing surroundings. *ICRA*, 2019. 2

[43] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*. Curran Associates, Inc., 2019. 6

[44] Jonah Philion and Sanja Fidler. Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. *ECCV*, 2020. 1, 2

[45] Alex D. Pon, Jason Ku, Chengyao Li, and Steven L. Waslander. Object-centric stereo matching for 3d object detection. *ICRA*, 2020. 1

[46] Rui Qian, Divyansh Garg, Yan Wang, Yurong You, Serge Belongie, Bharath Hariharan, Mark Campbell, Kilian Q. Weinberger, and Wei-Lun Chao. End-to-end pseudo-lidar for image-based 3d object detection. *CVPR*, 2020. 1

[47] Thomas Roddick and Roberto Cipolla. Predicting semantic map representations from images using pyramid occupancy networks. *CVPR*, 2020. 2

[48] Thomas Roddick, Alex Kendall, and Roberto Cipolla. Orthographic feature transform for monocular 3D object detection. *BMVC*, 2018. 1, 3, 6

[49] Samuel Schulter, Menghua Zhai, Nathan Jacobs, and Manmohan Chandraker. Learning to look around objects for top-view representations of outdoor scenes. *ECCV*, 2018. 2

[50] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. PV-RCNN: Point-voxel feature set abstraction for 3D object detection. *CVPR*, 2020. 1

[51] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. PointRCNN: 3D object proposal generation and detection from point cloud. *CVPR*, 2019. 1

[52] Andrea Simonelli, Samuel Rota Bulò, Lorenzo Porzi, Manuel López-Antequera, and Peter Kontschieder. Disentangling monocular 3d object detection. *ICCV*, 2019. 2, 6, 11

[53] Andrea Simonelli, Samuel Rota Bulò, Lorenzo Porzi, Elisa Ricci, and Peter Kontschieder. Towards generalization across depth for monocular 3d object detection. *ECCV*, 2020. 2, 6, 11

[54] Leslie N. Smith. A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay. *arXiv preprint*, 2018. 6

[55] Siddharth Srivastava, Frédéric Jurie, and Gaurav Sharma. Learning 2d to 3d lifting for object detection in 3d for autonomous vehicles. *arXiv preprint*, 2019. 3

[56] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset, 2019. 2, 6, 7

[57] Yunlei Tang, Sebastian Dorn, and Chiragkumar Savani. Center3d: Center-based monocular 3d object detection with joint depth understanding. *arXiv preprint*, 2020. 5, 8

[58] Lijun Wang, Jianming Zhang, Oliver Wang, Zhe Lin, and Huchuan Lu. Sdc-depth: Semantic divide-and-conquer network for monocular depth estimation. *CVPR*, 2020. 2

[59] 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. *CVPR*, 2019. 1, 3

[60] Ziyan Wang, Buyu Liu, Samuel Schulter, and Manmohan Chandraker. A parametric top-view representation of complex road scenes. *CVPR*, 2019. 2

[61] Xinshuo Weng and Kris Kitani. Monocular 3d object detection with pseudo-lidar point cloud. *ICCVW*, 2019. 3, 6, 11

[62] Bin Xu and Zhenzhong Chen. Multi-level fusion based 3D object detection from monocular images. *CVPR*, 2018. 3

[63] Dan Xu, Wanli Ouyang, Xiaogang Wang, and Nicu Sebe. Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing. *CVPR*, 2018. 2

[64] Tae-Kyun Kim Xuepeng Shi, Zhixiang Chen. Distance-normalized unified representation for monocular 3d object detection. *ECCV*, 2020. 3, 6, 11

[65] Xiaoqing Ye, Liang Du, Yifeng Shi, Yingying Li, Xiao Tan, Jianfeng Feng, Errui Ding, and Shilei Wen. Monocular 3d object detection via feature domain adaptation. *ECCV*, 2020. 3, 6

[66] Zhenyu Zhang, Zhen Cui, Chunyan Xu, Zequn Jie, Xiang Li, and Jian Yang. Joint task-recursive learning for semantic segmentation and depth estimation. *ECCV*, 2018. 2