Point Density-Aware Voxels for LiDAR 3D Object Detection

Jordan S. K. Hu  Tianshu Kuai  Steven L. Waslander
University of Toronto Robotics Institute
{jordan.hu, tianshu.kuai}@mail.utoronto.ca, steven.waslander@robotics.utias.utoronto.ca

Abstract

LiDAR has become one of the primary 3D object detection sensors in autonomous driving. However, LiDAR’s diverging point pattern with increasing distance results in a non-uniform sampled point cloud ill-suited to discretized volumetric feature extraction. Current methods either rely on voxelized point clouds or use inefficient farthest point sampling to mitigate detrimental effects caused by density variation but largely ignore point density as a feature and its predictable relationship with distance from the LiDAR sensor. Our proposed solution, Point Density-Aware Voxel network (PDV), is an end-to-end two stage LiDAR 3D object detection architecture that is designed to account for these point density variations. PDV efficiently localizes voxel features from the 3D sparse convolution backbone through voxel point centroids. The spatially localized voxel features are then aggregated through a density-aware RoI grid pooling module using kernel density estimation (KDE) and self-attention with point density positional encoding. Finally, we exploit LiDAR’s point density to distance relationship to refine our final bounding box confidences. PDV outperforms all state-of-the-art methods on the Waymo Open Dataset and achieves competitive results on the KITTI dataset. We provide a code release for PDV which is available at https://github.com/TRAILab/PDV.

1. Introduction

3D object detection is one of the key perception problems in the autonomous vehicle space as object pose estimation directly impacts the effectiveness of downstream tasks in the perception pipeline. Within the autonomous driving sensor stack, LiDAR has become one of the most popular sensors used for 3D object detection [24,26,40], because of the accurate 3D point cloud it produces through laser light.

However, the reliance on LiDAR data comes at the cost of point density variations across distance. Other factors such as occlusion play a role, but the primary reason is the natural divergence of points from the LiDAR with increasing distance due to the angular offsets between the Li-

DAR lasers. Thus, objects located at farther distances return fewer points than objects located closer to the LiDAR.

Voxel-based methods [5,40,43,46] typically ignore point density, solely relying on the quantized representation of the point cloud. When a high voxel resolution is afforded, as is the case on the KITTI dataset [7], voxel-based methods [44] have outperformed point-based and point-voxel-based methods. However, on datasets with larger input spaces such as the Waymo Open Dataset [30], the voxel resolution is limited due to memory constraints. Fine object details are therefore lost due to spatial misalignment between the voxel features and the point cloud as shown in Figure 1 (a), resulting in a degradation in performance.

Other methods [24,26] attempt to remedy point density variations through farthest point sampling (FPS) as seen in Figure 1 (b). Although effective at sampling locations on non-uniformly distributed point clouds, the computation scales poorly as a function of the number of points in the point cloud, increasing runtime and limiting the number of sampled points for second-stage proposal refinement.

Point density also affects detection of smaller objects such as pedestrians and cyclists. These objects have less surface area to intersect the LiDAR’s laser beams, resulting in poorer object localization. Perhaps informatively, current state-of-the-art methods have largely ignored detection
performance for pedestrians and cyclists, focusing solely on
the car or vehicle class [5,15,16,44]. As we move towards
datasets with higher environment coverage, it is necessary
for architectures to be scalable to larger input spaces and to
serve as a multi-class solution for 3D object detection.

We therefore propose Point Density-Aware Voxel net-
work (PDV) to resolve these identified issues by leveraging
voxel point centroid localization and feature encodings that
directly account for point density in multi-class 3D object
detection. We summarize our approach with the following
three contributions.

1. **Voxel Point Centroid Localization.** PDV partitions the
LiDAR points in each non-empty voxel to calculate a point
centroid for each voxel feature, as shown in Figure 1 (c).
By localizing the voxel features with point centroids for
second-stage proposal refinement, PDV uses the point den-
sity distributions to retain fine-grained position informa-
tion in the feature encodings without requiring an expensive
point cloud sampling method such as FPS.

2. **Density-Aware RoI Grid Pooling.** We augment region
of interest (RoI) grid pooling [24] to encode local point
density as an additional feature. First, we use kernel den-
sity estimation (KDE) [18,22] to encode local voxel fea-
ture density at each grid point ball query, followed by self-
attention [34] between grid points with a novel point density
positional encoding. Density-aware RoI grid pooling cap-
tures localized point density information in the context of
the whole region proposal for second-stage refinement.

3. **Density Confidence Prediction.** We further refine
our bounding box confidence predictions by using the fi-
nal bounding box centroid location and the number of raw
LiDAR points within the final bounding box as additional
features. Thus, we use the inherent relationship between
distance and point density established by LiDAR for more
informed confidence predictions.

PDV outperforms all current state-of-the-art methods
on the Waymo Open Dataset [30] with an increase of
+0.65%/+/1.25%, +0.53%/+/0.46%, and +0.49%/+/0.71%
on the vehicle, pedestrian, and cyclist LEVEL_1/LEVEL_2
mAPH classes, respectively, and achieves competitive per-
formance on the KITTI dataset [7].

2. Related Work

**Point-based LiDAR 3D Object Detection.** Point-based
methods use the raw point cloud to extract point-level fea-
tures for bounding box predictions. F-PointNet [19] applies
PointNet [20,21] on the point cloud, segmented via image-
based 2D object detections. PointRCNN [26] directly gen-
erates RoIs at the point-level through a PointNet++ [21]
backbone and uses point-level features for bounding box re-
finement. STD [42] proposes PointsPool for RoI feature ex-
traction while 3DSSD [41] adopts a new sampling strategy
on the raw point cloud to preserve enough interior points for
objects in the downsampled point cloud. Point-GNN [28]
constructs a graph using the raw point cloud and aggre-
rates node-level features to generate predictions. Point-
based methods utilize expensive point cloud sampling and
grouping, which inevitably require long inference times.

**Voxel-based LiDAR 3D Object Detection.** Voxel-based
methods divide the point cloud into a voxel grid to di-
rectly apply 3D and 2D convolutions for generating pre-
dictions [12,40,46]. CIA-SSD [43] adopts a light network
on the bird’s eye view (BEV) grid to extract robust spatial-
semantic features with a confidence rectification module for
better post-processing. Voxel-RCNN [5] proposes Voxel
RoI pooling to generate RoI features by aggregating voxel
features. VoTr [16] proposes a transformer-based 3D back-
bone as an alternative to the standard sparse convolution
layers [8,40]. The performance of voxel-based methods is
limited by the quantized point cloud as fine-grained point-
level information is lost from the voxelization process.

**Point-Voxel-based LiDAR 3D Object Detection.** Point-
voxel-based methods utilize both voxel and point represen-
tations of the point cloud. SA-SSD [9] uses an auxiliary net-
work during training that interpolates point-level features
from intermediate voxel layers. PV-RCNN++ [25] proposes
RoI grid pooling to effectively aggregate FPS-sampled keypoint
features on evenly spaced grids inside each bounding box
proposal. PV-RCNN++ [25] proposes a modified version of
FPS for faster point sampling and VectorPool aggregation
for RoI grid pooling. CT3D [23] constructs a cylindrical-
shaped RoI at each bounding box proposal. It adopts a
transformer-based encoder-decoder architecture to extract
RoI features directly from the nearby points without using
intermediate voxel features. Pyramid-RCNN [15] extends
the idea of RoI grid pooling over a single set of evenly
spaced grids to multiple sets of grids at different scales with
adaptive ball query radius at a significantly higher compu-
tation cost. Current point-voxel based methods do not ex-
plicitly account for variations in point cloud density within
each RoI and typically require long inference times due to
the dependence on point cloud sampling.

**Point Density Estimation.** KDE [18,22] estimates a prob-
ability distribution function of a random variable using a
finite set of samples and a chosen kernel function and
bandwidth. A couple methods have used KDE for fea-
ture encoding in point clouds. MC Convolution [10] uses a
Monte Carlo estimation of the convolution integral to han-
dle non-uniform sampled point clouds and uses KDE to
estimate the likelihood of a point within a local convolu-
tion. PointConv [38] also uses KDE, but estimates the like-
lihood of each sample with an additional feed-forward net-
work (FFN). Rather than restricting the density estimate for
reweighting, we use KDE as an additional feature within
each grid point ball query in density-aware RoI grid pool-
ing.
3. Methodology

PDV uses a two-stage approach with a 3D sparse convolution backbone for initial bounding box proposals which are then refined in a second stage through voxel features in each voxel layer and the raw point cloud data. Figure 2 shows an overview of the PDV framework.

3.1. 3D Voxel Backbone

We use a similar voxel backbone for initial bounding box proposals to SECOND [40]. The input to PDV is a point cloud which is defined as a set of 3D points \( \mathcal{P} = \{ \mathbf{x}_p, \mathbf{f}_p \} \) where \( \mathbf{x}_p \in \mathbb{R}^3 \) are the xyz spatial coordinates, \( \mathbf{f}_p \in \mathbb{R}^F \) are additional features such as the intensity or elongation of each point, and \( N_p \) is the number of points in the point cloud. First, the point cloud is voxelized and subsequently encoded using a series of 3D sparse convolutions [8, 40], followed by a region proposal network (RPN) for initial bounding box proposals. Each voxel layer has a different spatial resolution with 1x, 2x, 4x, and 8x downsampled resolutions based on the original voxel grid size. The voxel features in each layer are used for bounding box refinement in the second stage.

3.2. Voxel Point Centroid Localization

Inspired by grid-subsampling in KPConv [32, 33], the voxel point centroid localization module spatially locates non-empty voxel features for aggregation in density-aware RoI grid pooling.

Let \( \mathcal{V}_l = \{ \mathbf{V}_k^l = \{ \mathbf{h}_{\mathbf{V}_k^l}, \mathbf{f}_{\mathbf{V}_k^l} \} \mid k = 1, \ldots, N_l \} \) be the set of non-empty voxels in the \( l \)-th voxel layer where \( \mathbf{h}_{\mathbf{V}_k^l} \) is the 3D voxel index, \( \mathbf{f}_{\mathbf{V}_k^l} \) is the associated voxel feature vector, and \( N_l \) is the number of non-empty voxels for voxel layers \( l = 1, \ldots, L \). First, points that are within the same voxel are grouped together into a set \( \mathcal{N}(\mathbf{V}_k^l) \) by calculating their voxel index \( \mathbf{h}_{\mathbf{V}_k^l} \) from their spatial coordinates \( \mathbf{x}_t \) and voxel grid dimensions. The point centroid of each voxel feature is then calculated as

\[
c_{\mathbf{V}_k^l} = \frac{1}{|\mathcal{N}(\mathbf{V}_k^l)|} \sum_{\mathbf{x}_p \in \mathcal{N}(\mathbf{V}_k^l)} \mathbf{x}_p.
\]

Since the voxels in the convolution layers are in a sparse format, an intermediate hash table is used to efficiently map each calculated voxel point centroid to its corresponding feature vector. As shown in Figure 3, both voxel point centroids and sparse voxel features are associated with a shared voxel index. The intermediate hash table links the centroid \( c_{\mathbf{V}_k^l} \) with \( \mathbf{V}_k^l \) using the matching voxel index \( \mathbf{h}_{\mathbf{V}_k^l} \).

An advantage with using voxels is that we can use the previous voxel layer centroid calculations to efficiently compute the subsequent voxel point centroids based on the stride, padding, and kernel size of the convolution block. Let \( \mathcal{C}_{k+1} = \{ c_{\mathbf{V}_k^l} \mid K_{l+1}(\mathbf{h}_{\mathbf{V}_k^l}) = \mathbf{h}_{\mathbf{V}_{k+1}^l} \} \) be the set of voxel point centroids where \( K_{l+1} \) is the convolution block that maps voxel index \( \mathbf{h}_{\mathbf{V}_k^l} \) to \( \mathbf{h}_{\mathbf{V}_{k+1}^l} \). We can then perform a weighted average of the grouped voxel point centroids to...
calculate the centroids in the subsequent layer:
\[
c_{V_{l+1}}^k = \frac{1}{|N(V_{l+1}^k)|} \sum_{c_{V_{j}}^l \in c_{V_{l+1}}^k} |N(V_{j}^l)| c_{V_{j}}^l
\]  
(2)

where \( |N(V_{l+1}^k)| \) is calculated from
\[
|N(V_{l+1}^k)| = \sum_{c_{V_{j}}^l \in c_{V_{l+1}}^k} |N(V_{j}^l)|.
\]  
(3)

By avoiding recomputing centroids using the entire point cloud for each layer, voxel point centroid localization scales to larger point clouds more efficiently.

3.3. Density-aware RoI Grid Pooling

Density-aware RoI grid pooling builds upon RoI grid pooling [24] by augmenting the pooling method with a combination of KDE and self-attention to encode point density features into each proposal. First, \( U \times U \times U \) uniform grid points \( G^b = \{ g_1, \ldots, g_{U^3} \} \) are sampled for each bounding box proposal \( b \).

Local Feature Density. We use KDE to estimate local feature density within each grid point ball query. Rather than limiting the estimated density as a feature reweighting like MC Convolution [10] and PointConv [38], density-aware RoI grid pooling encodes the estimated probability density as an additional feature in the ball query for a more implicit feature encoding. First, we aggregate neighbouring features near each grid point where \( \mathcal{N}(g_j) \) is the set of voxel point centroids in a sphere of radius \( r \) centered around \( g_j \):
\[
\Psi_{g_j}^l = \left\{ \left[ f_{V_j}^l, c_{V_j}^l - g_j, p(c_{V_j}^l | g_j) \right] \right\}^T, \quad \forall c_{V_j}^l \in \mathcal{N}(g_j)
\]  
(4)

where the local offset \( c_{V_j}^l - g_j \) and likelihood \( p(c_{V_j}^l | g_j) \) are appended as additional features, as shown in Figure 4.

The likelihood is calculated for each grid point using KDE:
\[
p(c_{V_j}^l | g_j) \approx \frac{1}{|N(V_j^l)| \sigma^3} \sum_{c_{V_i}^l \in N(g_j)} W(c_{V_i}^l, c_{V_j}^l)
\]  
(5)

where \( \sigma \) is the bandwidth and \( W \) is
\[
W(c_{V_i}^l, c_{V_j}^l) = \prod_{d=1}^{3} w(d, c_{V_i}^{l:d} - c_{V_j}^{l:d})
\]  
(6)

with an independent kernel \( w \) on each xyz dimension \( d \). Once the features are appended, a PointNet multi-scale grouping (MSG) module [20, 21] is used to obtain a feature vector \( f_{g_j}^l \) for each grid point \( g_j \):
\[
f_{g_j}^l = \text{maxpool}(\text{FFN}(\Psi_{g_j}^l)).
\]  
(7)

We use multiple radii \( r \) to capture feature density at different scales for each grid point and concatenate the output features together. Finally, features are appended from different voxel layers to obtain the final features for each grid point:
\[
f_{g_j} = [f_{g_j}^1, \ldots, f_{g_j}^L].
\]  
(8)

Grid Point Self-Attention. The features encoded at each RoI grid point are localized to the size of the ball query but lack interdependent relationships between different grid points. An easy solution is to use self-attention [34] to capture long-range dependencies between the grid points but simply adding an attention module lacks the geometric information of the LiDAR point cloud. Thus, we also introduce a novel type of positional encoding that takes into consideration the point density within the point cloud.

As shown in Figure 5, the self-attention module performs self-attention between the non-empty grid point features \( f_{g_j}^b = \left\{ f_{g_i} \left| N(g_i) > 0, \forall g_i \in G^b \right. \right\} \) using a standard transformer encoder layer [34] and a residual connection similar to a non-local neural network block [35]:
\[
\tilde{f}_{g_j} = \mathcal{T}_{g_j}(f_{g_j}^b) + f_{g_j}.
\]  
(9)
where $T_{g_i}$ is the transformer encoder layer output for $f_{g_i}$, and $f_{g_i}$ is the output grid feature. Empty grid point features $|\mathcal{N}(g_i)| = 0$ are untouched by the self-attention module and are kept as their original feature encoding.

**Point Density Positional Encoding.** We add positional encoding to the self-attention module by using the local grid point positions and the number of points in the box proposal. The bounding box proposal is divided into voxels $V_{g_i}$ using the same $U \times U \times U$ grid resolution to establish voxels for each grid point. The positional encoding for each grid feature is then calculated as:

$$
\text{PE}(f_{g_i}) = \text{FFN} \left( \left[ \delta_{g_i}, \log \left( |\mathcal{N}(V_{g_i})| + \epsilon \right) \right] \right)
$$

where $\delta_{g_i} = x_{g_i} - c_b$ is the relative position of $g_i$ from the bounding box proposal centroid $c_b$, $|\mathcal{N}(V_{g_i})|$ is the number of points in each grid point voxel $V_{g_i}$, and $\epsilon$ is a constant offset. By leveraging the local offsets and number of points within each voxel, density-aware RoI grid pooling is able to capture point density variations within each region proposal.

### 3.4. Density Confidence Prediction

PDV also uses the relationship between the distance and the number of LiDAR points on scanned objects to predict the confidence of the final bounding box predictions. A shared FFN first encodes the flattened features from the density-aware RoI grid pooling module. Then, two separate FFN branches encode the features for the box refinement and box confidence outputs. In the box confidence branch, we additionally append two features to predict the output confidence $p_b$ for the final bounding box $b$:

$$
p_b = \text{FFN} \left( \left[ f^b, c_b, \log \left( |\mathcal{N}(b)| \right) \right] \right)
$$

where $f^b$ is the output feature vector from the shared FFN, $c_b$ is the centroid of the final bounding box, and $|\mathcal{N}(b)|$ is the number of raw points in the final bounding box.

### 3.5. Training Losses

We use an end-to-end training strategy for PDV with a region proposal loss $L_{\text{RPN}}$ and proposal refinement loss $L_{\text{RCNN}}$ that are trained jointly. The $L_{\text{RPN}}$ is calculated as:

$$
L_{\text{RPN}} = L_{\text{cls}}(y_b, y^*_b) + \beta L_{\text{reg}}(r_b, r^*_b)
$$

where $L_{\text{cls}}$ is the focal loss [13], $L_{\text{reg}}$ is the smooth-L1 loss, $y_b$ is the predicted class vector, $y^*_b$ is the ground truth class, $r_b$ is the predicted RoI anchor residual, $r^*_b$ is the ground truth anchor residual, and $\beta$ is a scaling factor. $L_{\text{RCNN}}$ is composed of:

$$
L_{\text{RCNN}} = L_{\text{IoU}} + L_{\text{reg}}(r_b, r^*_b)
$$

where $r_b$ is the predicted bounding box residual and $r^*_b$ is the ground truth residual. A smooth-L1 loss is used to regress the bounding box residuals. We use the same confidence and regression targets as PV-RCNN [24].

### 4. Experimental Results

**Datasets.** We evaluate PDV on the Waymo Open Dataset [30] and the KITTI 3D Object Detection benchmark [7]. The Waymo Open Dataset is one of the biggest and most diverse autonomous driving datasets available, containing 798 training sequences (approximately 158k point cloud samples) and 202 validation sequences (approximately 40k point cloud samples), with annotations for objects in full 360° field of view. The Waymo Open Dataset uses standard mean average precision (mAP) as well as mAPH, which factors in heading angle. The predictions are split into LEVEL_1, which only includes 3D labels with more than five LiDAR points, and LEVEL_2, which also includes 3D labels with at least one LiDAR point. The KITTI dataset contains 7,481 training samples and 7,518 testing samples and uses standard average precision (AP) on easy, moderate and hard difficulties. We adopt the standard training and val split sets from Chen et al. [4].

**Input Parameters.** For the Waymo Open Dataset [30], the detection range is [-75.2m, 75.2m] for the X and Y axes, and [-2m, 4m] for the Z axis. We divide the raw point cloud into voxels of size (0.1m, 0.1m, 0.15m). Since the KITTI dataset [7] only provides annotations in front camera’s field of view, its detection range is set to be [0, 70.4m] for the X axis, [-40m, 40m] for the Y axis, and [-3m, 1m] for the Z axis. We set the voxel size to be (0.05m, 0.05m, 0.1m).

**Network Architecture.** PDV uses the last two voxel layers $l = 3, 4$ for voxel point centroid localization. Density-aware RoI grid pooling uses a grid size of $U = 6$. Each
PDV performs well on the 3D mAPH, respectively, compared to PV-RCNN. We also outperform PV-RCNN++ [25] by +1.25%, +0.46%, and +0.71% on vehicle, pedestrian, and cyclist LEVEL 2 mAP, respectively, compared to PV-RCNN. We also outperform Pillar-OD [36] by +0.02%, +0.13%, and +0.17% on vehicle, pedestrian, and cyclist LEVEL 2 mAP, respectively.

Table 2 shows PDV's performance across distance on the validation set in Table 1. PDV achieves state-of-the-art results on all classes on both LEVEL 1 and LEVEL 2 mAP/mAPH metrics. We outperform methods that use PV-RCNN as a base architecture [15, 16, 24] by at least +2.13% and +1.88% on pedestrian and cyclist LEVEL 2 mAP, respectively, compared to PV-RCNN. We also outperform Pillar-OD [36] by +0.02%, +0.13%, and +0.17% on vehicle, pedestrian, and cyclist LEVEL 2 mAP, respectively.

4.1. Waymo Dataset Results

We report the multi-class Waymo Open Dataset results on the validation set in Table 1. PDV achieves state-of-the-art results on all classes on both LEVEL 1 and LEVEL 2 mAP/mAPH metrics. We outperform methods that use PV-RCNN as a base architecture [15, 16, 24] by at least +2.13% and +1.88% on pedestrian and cyclist LEVEL 2 3D mAPH, respectively, compared to PV-RCNN. We also outperform PV-RCNN++ [25] by +1.25%, +0.46%, and +0.71% on vehicle, pedestrian, and cyclist LEVEL 2 3D mAPH, respectively. PDV effectively captures fine point details lost in the voxel backbone through point density for accurate bounding box refinement in the second stage.

Table 2 shows PDV’s performance across distance on the vehicle, pedestrian, and cyclist LEVEL 2 mAP, respectively.
Table 4. 3D detection results on the KITTI val set for car, pedestrian, and cyclist classes using AP[40]. Bolded and underlined values are best and second-best performance, respectively.

| Method          | Car 3D (IoU = 0.7) | Pedestrian 3D (IoU = 0.5) | Cyclist 3D (IoU = 0.5) |
|-----------------|---------------------|---------------------------|------------------------|
|                 | Easy | Moderate | Hard  | Easy | Moderate | Hard  | Easy | Moderate | Hard  |
| PDV (Ours)      | 92.56 | 85.29 | 83.05 | 66.90 | 60.80 | 55.85 | 92.72 | 74.23 | 69.60 |
| Improvement     | +0.22 | +0.32 | +0.14 | +2.64 | +4.13 | +3.94 | +3.71 | +2.28 | +1.69 |

Table 5. 3D detection results on the KITTI test set for car and cyclist using AP[41]. Bolded and underlined values are best and second-best performance, respectively.

| Method          | Car 3D (IoU = 0.7) | Cyclist 3D (IoU = 0.5) |
|-----------------|---------------------|------------------------|
|                 | Easy | Moderate | Hard  | Easy | Moderate | Hard  |
| SECOND [40]     | 83.34 | 72.55 | 65.82 | 71.33 | 52.08 | 45.83 |
| PointPillar [12]| 82.58 | 74.31 | 68.99 | 77.10 | 58.65 | 51.92 |
| STD [42]        | 87.95 | 79.71 | 75.09 | 78.69 | 61.59 | 55.30 |
| 3DSSD [43]      | 88.36 | 79.57 | 74.55 | 82.48 | 64.10 | 56.90 |
| SA-SSD [9]      | 88.75 | 79.79 | 74.16 | -     | -     | -     |
| SE-SSD [44]     | 91.49 | 82.54 | 77.15 | -     | -     | -     |
| PV-RCNN [24]    | 90.25 | 81.43 | 76.82 | 78.60 | 63.71 | 57.65 |
| PV-RCNN++ [25]  | 90.14 | 81.88 | 77.15 | 82.22 | 67.33 | 60.04 |
| Voxel-RCNN [5]  | 90.90 | 81.62 | 77.06 | -     | -     | -     |
| DSS-PV [1]      | 88.25 | 81.46 | 76.96 | 82.19 | 68.54 | 61.33 |
| CT3D [23]       | 87.83 | 81.77 | 77.16 | -     | -     | -     |
| VoTr-TSD [16]   | 89.90 | 82.09 | 79.14 | -     | -     | -     |
| Pyramid-PV [15] | 88.39 | 82.08 | 77.49 | -     | -     | -     |
| PDV (Ours)      | 90.43 | 81.86 | 77.36 | 83.04 | 67.81 | 60.46 |

4.3. Ablation Studies

We provide ablation studies to study the effects of each component in PDV. All models and variations are trained on 10% of the Waymo training dataset for 60 epochs.

Components. Table 6 shows the relative performance gain of each component on LEVEL_2 mAPH on the Waymo Open Dataset validation set. Experiment 1 uses voxel centers to localize voxel features. By localizing voxel features using voxel point centroids instead, Experiment 2 provides an improvement of +0.83%, +3.86%, and +3.67% for the vehicle, pedestrian, and cyclist classes, respectively. Voxel feature localization is extremely beneficial for smaller objects since voxel centers do not have proper alignment with the point cloud. By localizing the features closer to the object’s scanned surface, voxel point centroids contain meaningful geometric shape information for proposal refinement.

Local feature density estimation improves scores by +0.07%, 4.45%, +1.78% as shown in Experiment 3. By capturing feature density relationships via KDE, local feature density estimation provides a large improvement for the deformable pedestrian class, where encoded features have variable spatial configurations. Experiment 4 shows attention improves results by +0.45%, +0.61%, +1.43% providing a large increase in the pedestrian and cyclist class by establishing long-range dependencies between RoI grid points. Finally, Experiment 5 shows the effect of density confidence prediction, improving performance on the pedestrian and cyclist classes by +0.23% and +0.74%, respectively, with a small drop on the vehicle class of -0.01%.

Point Density Positional Encoding. Table 7 shows the effects of point density positional encoding in density-aware RoI pooling. Experiment 1 shows the baseline performance without any positional encoding. Experiment 2 uses a sinusoidal encoding similar to DETR [3, 34] where each spatial dimension is encoded with independent sine and cosine functions at different frequencies, showing an improvement on the pedestrian and cyclist by +0.14% and +0.50%, but a reduction on vehicle by -0.25%. A FFN with spatial grid point coordinates as input provides an overall increase in performance of +0.17%, +0.18%, and +0.27% on vehicles, pedestrians, and cyclists, respectively, as shown in Experiment 3. In Experiment 4, the density feature, in the form of

| Method          | Car 3D (IoU = 0.7) | Pedestrian 3D (IoU = 0.5) | Cyclist 3D (IoU = 0.5) |
|-----------------|---------------------|---------------------------|------------------------|
|                 | Easy | Moderate | Hard  | Easy | Moderate | Hard  | Easy | Moderate | Hard  |
| SECOND [40]     | 83.34 | 72.55 | 65.82 | 71.33 | 52.08 | 45.83 |
| PointPillar [12]| 82.58 | 74.31 | 68.99 | 77.10 | 58.65 | 51.92 |
| STD [42]        | 87.95 | 79.71 | 75.09 | 78.69 | 61.59 | 55.30 |
| 3DSSD [43]      | 88.36 | 79.57 | 74.55 | 82.48 | 64.10 | 56.90 |
| SA-SSD [9]      | 88.75 | 79.79 | 74.16 | -     | -     | -     |
| SE-SSD [44]     | 91.49 | 82.54 | 77.15 | -     | -     | -     |
| PV-RCNN [24]    | 90.25 | 81.43 | 76.82 | 78.60 | 63.71 | 57.65 |
| PV-RCNN++ [25]  | 90.14 | 81.88 | 77.15 | 82.22 | 67.33 | 60.04 |
| Voxel-RCNN [5]  | 90.90 | 81.62 | 77.06 | -     | -     | -     |
| DSS-PV [1]      | 88.25 | 81.46 | 76.96 | 82.19 | 68.54 | 61.33 |
| CT3D [23]       | 87.83 | 81.77 | 77.16 | -     | -     | -     |
| VoTr-TSD [16]   | 89.90 | 82.09 | 79.14 | -     | -     | -     |
| Pyramid-PV [15] | 88.39 | 82.08 | 77.49 | -     | -     | -     |
| PDV (Ours)      | 90.43 | 81.86 | 77.36 | 83.04 | 67.81 | 60.46 |
number of points within each grid point voxel, shows similar improvements over the baseline with +0.22%, +0.73%, and +0.67%. When we combine both local spatial coordinates and density in Experiment 5, we obtain the best performance with increases of +0.15%, +1.07%, +1.82%.

### 4.4. Runtime Analysis

Table 8 shows the runtime comparisons of PDV. We train the models for 60 epochs on 10% of the training data for Waymo and 80 epochs on the training set for KITTI. Each model is evaluated with an Intel i7-6850K processor, a single Titan Xp GPU, and a batch size of 1. For a fair comparison, we use the same number of $K$ proposals from the RPN ($K = 275$ for Waymo and $K = 100$ for KITTI) for all models. PDV outperforms PV-RCNN on inference speed and performance, reducing runtime by 14% and 5% for Waymo and KITTI, respectively, and improving mAPH by +2.42% and mAP by +1.51% on Waymo and KITTI, respectively.

Table 8 also shows the issue with voxel-based methods on larger input spaces. Although Voxel-RCNN is computationally efficient and performs well on the KITTI dataset, its performance degrades significantly, performing worse than PV-RCNN by -0.67% on Waymo. On the other hand, voxel point centroids provide an efficient alternative that retains spatial fidelity across the two datasets. Using voxel point centroids only (VC Only) results in an efficient architecture with a decrease in runtime by 61% and better performance on LEVEL.2 mAPH by +1.22% compared to PV-RCNN on Waymo. VC Only also scales to the large Waymo dataset, with only an increase of 64 ms in runtime from KITTI, showing voxel point centroid localization as an efficient alternative to FPS for spatially locating voxel features. PDV overcomes the limitations of voxel-based methods when scaling to larger datasets, while retaining computational efficiency.

### 4.5. False Positives across Distance

Figure 6 shows the number of false positives (IoU $< 0.7$) predicted by PDV and PV-RCNN across distance for vehicles on the Waymo dataset. As the distance from the sensor increases, the gap between PDV and PV-RCNN increases. We attribute the increased differential to using point density to refine bounding box regression and confidence values, which is more beneficial at detecting challenging objects at farther ranges.

### 5. Conclusion

We present PDV, a novel LiDAR 3D object detection method that uses voxel features and raw point cloud data to account for point density variations in LiDAR point clouds. PDV is particularly useful on large input spaces where point cloud sampling is expensive and voxel resolutions are low, resulting in state-of-the-art performance on the Waymo dataset and competitive results for the KITTI dataset.
References

[1] Prarthana Bhattacharyya, Chengjie Huang, and Krzysztof Czarnecki. Sa-det3d: Self-attention based context-aware 3d object detection. arXiv preprint, 2021. 7
[2] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. CVPR, 2020. 11
[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. ECCV, 2020. 7
[4] Xiaozhi Chen, Kaustav Kundu, Yukun Zhu, Andrew G Berneshawi, Huimin Ma, Sanja Fidler, and Raquel Urtasun. 3d object proposals for accurate object class detection. NeurIPS, 2015. 5
[5] Jiajun Deng, Shaoshuai Shi, Peiwei Li, Wengang Zhou, Yanyong Zhang, and Houqiang Li. Voxel r-cnn: Towards high performance voxel-based 3d object detection. AAAI, 2021. 1, 2, 6, 7, 8
[6] Runzhou Ge, Zhanguzhung Ding, Yihan Hu, Yu Wang, Sijia Chen, Li Huang, and Yuan Li. Afdet: Anchor free one stage 3d object detection. arXiv preprint, 2020. 6
[7] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. CVPR, 2012. 1, 2, 5, 6, 11
[8] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. CVPR, 2018. 2, 3
[9] Chenhang He, Hui Zeng, Jianqiang Huang, Xian-Sheng Hua, and Lei Zhang. Structure aware single-stage 3d object detection from point cloud. CVPR, 2020. 2, 7
[10] Pedro Hermosilla, Tobias Ritschel, Pere-Pau Vázquez, Álvar Vinacua, and Timo Ropinski. Monte carlo convolution for learning on non-uniformly sampled point clouds. ACM Transactions on Graphics (TOG), 2018. 2, 4
[11] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint, 2014. 6
[12] 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, 6, 7
[13] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. CVPR, 2017. 5
[14] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint, 2016. 6
[15] Jiageng Mao, Minzhe Niu, Haoyue Bai, Xiaodan Liang, Hang Xu, and Chunjing Xu. Pyramid r-cnn: Towards better performance and adaptability for 3d object detection. ICCV, 2021. 2, 6, 7
[16] Jiageng Mao, Yujing Xue, Minzhe Niu, Haoyue Bai, Jiashi Feng, Xiaodan Liang, Hang Xu, and Chunjing Xu. Voxel transformer for 3d object detection. ICCV, 2021. 2, 6, 7
[17] Xuran Pan, Zhuhuan Xia, Shiji Song, Li Erran Li, and Gao Huang. 3d object detection with pointformer. CVPR, 2021. 6
[18] Emanuel Parzen. On Estimation of a Probability Density Function and Mode. The Annals of Mathematical Statistics, 1962. 2
[19] Charles R. Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum pointnets for 3d object detection from rgb-d data. CVPR, 2018. 2
[20] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. CVPR, 2017. 2, 4
[21] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. NeurIPS, 2017. 2, 4
[22] Murray Rosenblatt. Remarks on Some Nonparametric Estimates of a Density Function. The Annals of Mathematical Statistics, 1956. 2
[23] Hualiang Sheng, Sijia Cai, Yuan Liu, Bing Deng, Jianqiang Huang, Xian-Sheng Hua, and Min-Jian Zhao. Improving 3d object detection with channel-wise transformer. ICCV, 2021. 2, 6, 7, 11, 12
[24] 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, 2, 4, 5, 6, 7, 8, 11, 12, 13
[25] Shaoshuai Shi, Li Jiang, Jiajun Deng, Zhe Wang, Chaoxu Guo, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn++: Point-voxel feature set abstraction with local vector representation for 3d object detection. arXiv preprint, 2021. 2, 6, 7, 12
[26] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. CVPR, 2019. 1, 2
[27] Shaoshuai Shi, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network. TPAMI, 2020. 6
[28] Weijing Shi and Raj Rajkumar. Point-gnn: Graph neural network for 3d object detection in a point cloud. CVPR, 2020. 2
[29] 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
[30] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. CVPR, 2020. 1, 2, 5, 6, 11
[31] OpenPCDet Development Team. Openpcdet: An open-source toolbox for 3d object detection from point clouds. https://github.com/open-mmlab/OpenPCDet, 2020. 7, 8, 12
[32] Hugues Thomas, François Goulette, Jean-Emmanuel Deschaud, Beatriz Marcotegui, and Yann LeGall. Semantic classification of 3d point clouds with multiscale spherical neighborhoods. 3DV, 2018. 3
[33] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. ICCV, 2019. 3
[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. NeurIPS, 2017. 2, 4, 7

[35] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. CVPR, 2018. 4

[36] Yue Wang, Alireza Fathi, Abhijit Kundu, David A Ross, Caroline Pantofaru, Tom Funkhouser, and Justin Solomon. Pillar-based object detection for autonomous driving. ECCV, 2020. 6

[37] Ze Wang, Sihao Ding, Ying Li, Jonas Fenn, Sohini Roychowdhury, Andreas Wallin, Lane Martin, Scott Ryvola, Guillermo Sapiro, and Qiang Qiu. Cirrus: A long-range bi-pattern lidar dataset. ICRA, 2021. 11

[38] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. CVPR, 2019. 2, 4

[39] Qiangeng Xu, Yin Zhou, Weiyue Wang, Charles R Qi, and Dragomir Anguelov. Spg: Unsupervised domain adaptation for 3d object detection via semantic point generation. ICCV, 2021. 11

[40] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. Sensors, 2018. 1, 2, 3, 6, 7

[41] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3dssd: Point-based 3d single stage object detector. CVPR, 2020. 2, 7

[42] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Std: Sparse-to-dense 3d object detector for point cloud. ICCV, 2019. 2, 7

[43] Wu Zheng, Weiliang Tang, Sijin Chen, Li Jiang, and Chi-Wing Fu. Cia-ssd: Confident iou-aware single-stage object detector from point cloud. AAAI, 2021. 1, 2

[44] Wu Zheng, Weiliang Tang, Li Jiang, and Chi-Wing Fu. Se-ssd: Self-ensembling single-stage object detector from point cloud. CVPR, 2021. 1, 2, 7

[45] Yin Zhou, Pei Sun, Yu Zhang, Dragomir Anguelov, Jiyang Gao, Tom Ouyang, James Guo, Jiquan Ngiam, and Vijay Vasudevan. End-to-end multi-view fusion for 3d object detection in lidar point clouds. CoRL, 2020. 6

[46] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. CVPR, 2018. 1, 2
In the supplementary material, we provide a discussion on the potential negative impact and limitations of PDV in Appendix A and Appendix B, respectively. We also provide additional detailed results on the Waymo Open Dataset [30] in Appendix C as well as on the KITTI dataset [7] in Appendix D. Finally, we provide visualizations of the point density variations across distance in Appendix E.

A. Potential Negative Impact

As autonomous driving progresses towards public use, PDV has the potential to be used for surveillance of civilians. Privacy is a large concern in the digital age, and it is important to ensure that this information is not used without consent or to maliciously track people’s location.

Unlike other domains where safety concerns are minimal, reliable 3D object detection in autonomous driving is extremely important. Objects that are missed or even misclassified could have detrimental effects in subsequent decision-making tasks for the vehicle, running a risk for both the passengers and other individuals sharing the road. It is important to take the results of 3D object detection methods on improved benchmarks as only one aspect for reliable 3D object detection as increased performance on a metric is not indicative of the practicality of a method in a real-world scenario.

B. Limitations

Voxel Resolution. PDV shows the most improvement compared to other methods on large input spaces when the voxel resolution is limited, such as on the Waymo Open Dataset [30]. However, when the voxel resolution is sufficiently high, the improvements from PDV are less significant, as is the case on the KITTI dataset [7]. At higher voxel resolutions, the difference between the voxel center and the voxel point centroids decreases, resulting in less performance gains. Additionally, at high voxel resolutions, each occupied voxel represents a decent approximation of the point density, which can be captured through standard convolutions. Thus, PDV is most suited for when low voxel resolutions are necessary, such as for 360° detection, where the point density feature can be fully exploited for better performance.

LiDAR Dependency. PDV uses point density as an additional encoding feature, and therefore relies on the LiDAR specifications for accurate 3D object detection. Although PDV is tested on the KITTI dataset [7] and the Waymo Open Dataset [30], which have different LiDAR characteristics, it is necessary to expand other datasets to ensure that PDV is robust to different LiDAR sampling patterns. Some examples include Nuscenes [2], which uses a 32 beam LiDAR, and Cirrus [37], which uses a LiDAR designed for long range detection.

| Method     | Pedestrian LEVEL 1 mAP/mAPH |
|------------|----------------------------|
|            | 0-30m | 30-50m | 50m-Inf |
| PDV (Ours) | 80.32 | 73.60  | 61.69/50.07 |

Table 9. Performance comparison on the Waymo Open Dataset with 202 validation sequences for 3D pedestrian (IoU = 0.5) detection across distance.

| Method     | Cyclist LEVEL 1 mAP/mAPH |
|------------|-------------------------|
|            | 0-30m | 30-50m | 50m-Inf |
| PDV (Ours) | 80.86 | 62.61/61.45 | 46.23/44.12 |

Table 10. Performance comparison on the Waymo Open Dataset with 202 validation sequences for 3D cyclist (IoU = 0.5) detection across distance.

Adverse Weather Conditions. PDV uses point density to provide additional encoding in the second stage. If there is a significant distribution shift in the LiDAR point density during inference time, there may be significant degradation in performance. For example, as highlighted in SPG [39], the Waymo Kirkland dataset [30] has a different distribution of points, where rainy weather results in a significant number of points missing on objects. Since PDV relies on point density, a sudden shift in point density variations due to weather may cause severe degradation in performance. A future improvement to PDV is to ensure robustness to potential shifts in point density distributions during inference.

C. More Waymo Dataset Results

We show the Waymo Open Dataset validation results for the pedestrian and cyclist classes across distance for PDV using the first LiDAR return only in Table 9 and Table 10, respectively. The distance evaluation is broken down into three categories: 0 to 30 meters, 30 to 50 meters, and beyond 50 meters. We also show the multi-class results for PDV using first and second LiDAR return in Table 11. Overall, PDV utilizes the additional points from the second LiDAR return better than PV-RCNN++, providing the largest performance increase on the cyclist class.

D. More KITTI Dataset Results

We provide KITTI dataset results on the val set on 3D and BEV for AP|_{R_{11}} and AP|_{R_{40}} in Table 12, Table 13, and Table 14. PDV’s performance is compared to PV-RCNN [24] and CT3D [23].
Table 11. Performance comparison on the Waymo Open Dataset with 202 validation sequences for 3D vehicle (IoU = 0.7), pedestrian (IoU = 0.5) and cyclist (IoU = 0.5) detection using first and second LiDAR return.

| Method         | Veh. (LEVEL_1) mAP | Veh. (LEVEL_2) mAP | Ped. (LEVEL_1) mAP | Ped. (LEVEL_2) mAP | Cyc. (LEVEL_1) mAP | Cyc. (LEVEL_2) mAP |
|----------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|
| PV-RCNN† [24]  | 77.51              | 68.41              | 68.98              | 68.41              | 67.81              | 68.41              |
| PV-RCNN++ [25] | 78.79              | 70.26              | 67.21              | 65.71              | 68.51              | 66.04              |
| PDV (Ours)     | 79.43              | 70.47              | 69.98              | 68.41              | 71.45              | 69.11              |
| Improvement    | +0.64              | -0.44              | +0.27              | +0.27              | +2.47              | +2.55              |

Table 12. 3D detection results on the KITTI val set for car, pedestrian, and cyclist classes using AP$_{R^{11}}$. †: Results are taken from publicly released models [23, 24, 31].

| Method         | Car 3D (R$_{11}$) | Pedestrian 3D (R$_{11}$) | Cyclist 3D (R$_{11}$) |
|----------------|-------------------|--------------------------|-----------------------|
|                | Easy          Moderate   Hard      | Easy          Moderate   Hard      | Easy          Moderate   Hard      |
| PV-RCNN* [24]  | 89.35  83.69  78.70     | 64.60  57.90  53.23     | 85.22  70.47  65.75     |
| CT3D* [23]     | 89.11  85.04  78.76     | 61.74  56.28  52.51     | 85.04  71.71  68.05     |
| PDV (Ours)     | 89.52  84.03  79.09     | 65.83  61.18  55.87     | 90.48  73.23  69.55     |
| Improvement    | +0.41  -1.01  +0.33     | +1.23  +3.28  +2.64     | +5.26  +1.52  +1.50     |

Table 13. BEV detection results on the KITTI val set for car, pedestrian, and cyclist classes using AP$_{R^{40}}$. †: Results are taken from publicly released models [23, 24, 31].

| Method         | Car BEV (R$_{40}$) | Pedestrian BEV (R$_{40}$) | Cyclist BEV (R$_{40}$) |
|----------------|-------------------|--------------------------|-----------------------|
|                | Easy          Moderate   Hard      | Easy          Moderate   Hard      | Easy          Moderate   Hard      |
| PV-RCNN* [24]  | 90.09  87.90  87.41     | 67.01  61.38  56.10     | 86.79  73.55  69.69     |
| CT3D* [23]     | 90.25  88.18  87.78     | 64.23  59.84  55.76     | 90.94  73.68  71.21     |
| PDV (Ours)     | 90.33  88.33  87.91     | 69.01  63.54  59.46     | 90.77  73.75  71.21     |
| Improvement    | +0.08  +0.15  +0.13     | +2.00  +2.16  +3.36     | -0.17  +0.07  +0.00     |

Table 14. BEV detection results on the KITTI val set for car, pedestrian, and cyclist classes using AP$_{R^{40}}$. †: Results are taken from publicly released models [23, 24, 31].

E. Point Density Distance Plots

Figure 7 shows point density across distance plots for PV-RCNN [24] and PDV. By using the relationship between distance and point density (number of points within each final bounding box prediction), PDV effectively reduces the number of false positives outside the distribution of training samples across distance.

12
Figure 7. From left to right column: Number of points in ground truth boxes across distance for cars, pedestrians and cyclists on the KITTI dataset. The first row shows the distribution of training samples on the train split. The second and third row show the predictions of PV-RCNN [24] and PDV on the KITTI val split, respectively. Blue predictions are true positives while orange predictions are false positives for cars (IoU < 0.7), pedestrians (IoU < 0.5), and cyclists (IoU < 0.5).