It's All Around You: Range-Guided Cylindrical Network for 3D Object Detection

Meytal Rapoport-Lavie
Tel-Aviv University
laviemeytal@mail.tau.ac.il

Dan Raviv
Tel-Aviv University
darav@tauex.tau.ac.il

Abstract

Modern perception systems in the field of autonomous driving rely on 3D data analysis. LiDAR sensors are frequently used to acquire such data due to their increased resilience to different lighting conditions. Although rotating LiDAR scanners produce ring-shaped patterns in space, most networks analyze their data using an orthogonal voxel sampling strategy. This work presents a novel approach for analyzing 3D data produced by 360-degree depth scanners, utilizing a more suitable coordinate system, which is aligned with the scanning pattern. Furthermore, we introduce a novel notion of range-guided convolutions, adapting the receptive field by distance from the ego vehicle and the object's scale. Our network demonstrates powerful results on the competitive nuScenes 3D object detection challenge, comparable to current state-of-the-art architectures.

Figure 1. **Top-View of a nuScenes dataset sample with Ground Truth detections.** First, it can be easily observed that the LiDAR sensor has a ring-shaped pattern output. Second, the Cylindrical Coordinate system maintains 9 neighbors and 27 neighbors per voxel (including itself), in 2D and 3D respectively, the same as the classical Cartesian system. This enables us to employ the classical convolution layers without almost any alteration.

1. Introduction

Robustness is a crucial requirement for visual perception solutions in autonomous driving systems. Such systems must be resilient to various lighting scenarios and withstand harsh weather conditions without compromising their performance. Therefore, extending the traditional RGB cameras with additional sensors, such as LiDAR, is a vital step towards achieving this goal.

The inclusion of depth information, which allows capturing the three-dimensional structure of the vehicle's environment, is a key feature for ensuring robustness while maintaining high accuracy. Modern LiDAR acquisition sensors provide meaningful information not only for avoiding imminent collisions, but also to perceive the environment as good as image-based data and even surpass it under poor lighting conditions.

In the field of autonomous driving, there is a variety of methods to handle point clouds. The most common and efficient approach is to transform the point cloud into a regular representation, such as 2D bird-eye-view (BEV) voxels [13, 7, 27, 14] or 3D voxels [24, 31, 8, 26]. Following the architecture of PointNet and PointNet++ [4, 21], some works have proposed replacing the voxel-based method with a point-based one. This approach, however, was found to be computationally intensive. Recent papers have suggested combining both point-based and voxel-based methods into a single network [23, 16]. However, whether combining point-based methods or not, in 2D or 3D, these methods use the Cartesian coordinate system as their grid.

In this paper, we argue that projecting the points onto a Cartesian space disregards the circular nature of the raw

data provided by a rotating emitter as can be easily seen in Figure 1. We show that in order to perceive and preserve the additional information gained from a LiDAR sensor, one may leverage a more appropriate coordinate system, namely the Cylindrical coordinates system. Furthermore, the Cylindrical coordinate system allows us to align the entire network towards the sensor's point of view. Moreover, the closer an object is to the ego vehicle, the more points correspond to it in the point cloud. Therefore, we suggest adjusting the size of the voxels to the distance from the sensor, as is done naturally by the Cylindrical space. As shown in Figure 2, using the cylindrical system, we get a better distribution of average points per voxel over distance, leading to a better resolution after the voxelization step. In addition, we propose a novel guiding unit, which orchestrates convolutions by their range from the ego vehicle. We further explore the challenges for a "self-oriented" network to learn angle and velocity information and present the necessary solutions to overcome them successfully.

Our main contributions are as follows:

- We present the first end-to-end cylindrical coordinates perception architecture for autonomous systems that includes orientation and velocity estimation.
- We present a novel range-guided convolution block to adjust the network's receptive field according to an object's range and scale, and adapting its learned features by its distance from the ego-vehicle.
- We provide new insights, challenges and solutions for training deep networks in cylindrical coordinates.
- Our network achieves comparable results to current state-of-the-art networks on the competitive nuScenes 3D object detection benchmark.

2. Related works

Cartesian Based 3D Object Detection Methods: Most state-of-the-art methods for 3D object detection project the point clouds onto a Cartesian coordinate grid. Earlier approaches employed the BEV projection to utilize efficient 2D convolution layers to process the point cloud data [13, 7]. In a later work, VoxelNet [31] was first to introduce 3D voxels in the field of object detection, whereas SECOND [26] improved this method by applying sparse 3D convolutions to accelerate running time. The majority of cutting edge methods [24, 23], including nuScenes' former state-of-the-art, CBGS [32], have adopted the practice of processing point cloud using sparse three-dimensional convolutions after projecting it into the traditional Cartesian voxel space.

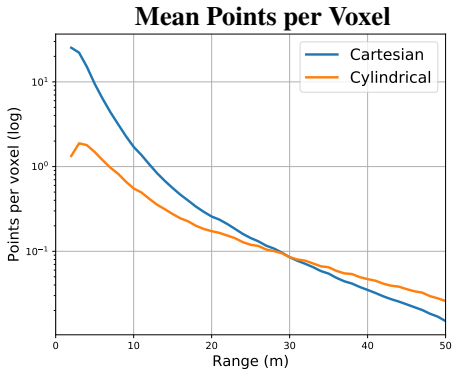

Figure 2. **Mean number of points per voxel (logarithmic) as a function of range.** There is a high inverse correlation between the number of points and their range from the sensor. Since the Cartesian voxels are of the same size in all ranges, we lose precision in the nearby distance. On the other hand, with Cylindrical coordinate we receive a better spatial distribution which leads to a better resolution after the voxelization step.

Non-Cartesian Based Methods: PolarNet [30] was the first paper to suggest quantizing LiDAR data in a non-Cartesian coordinate system. Their Polar-BEV network, designed for semantic segmentation on the SemanticKITTI [1] challenge, have shown improvement with respect to traditional Cartesian methods. Unlike semantic segmentation tasks, 3D detection challenges, such as nuScenes, require the processing of additional object orientation and velocity features. As direction information plays a significant role in such tasks, they are directly affected by the change in the coordinate system and therefore need to be treated with special care. A recently published paper, CVCNet [22], also quantizes the input LiDAR point cloud utilizing the polar grid, and then divide it into two separate views - BEV voxels (r, θ) and range-view voxels (θ, ϕ). Although the Polar coordinate system (r, θ, ϕ), is different than the cylindrical one (ρ, θ, z), it shares some of the challenges, when comparing both to the Cartesian system. Nevertheless, unlike our paper, the authors overlooked issues regarding the orientation and velocity predictions, and the change in the receptive field caused by the change in voxel size.

3D Object Detection Methods with Focus on Range: LaserNet [19] has introduced the concept of range-image input to the field of 3D detection, naturally emphasizing an object's distance from the ego vehicle as one of its key features. Despite its computational efficiency, it was generally outperformed by voxel-based methods. A later publication [2] suggested a method to address range-image's problem of scale variance, the fact that near distance objects appear larger in the image view. They applied dilated convolution [5, 25] directly to the range-image input, adjusting each pixel's dilation rate as a function of its range.

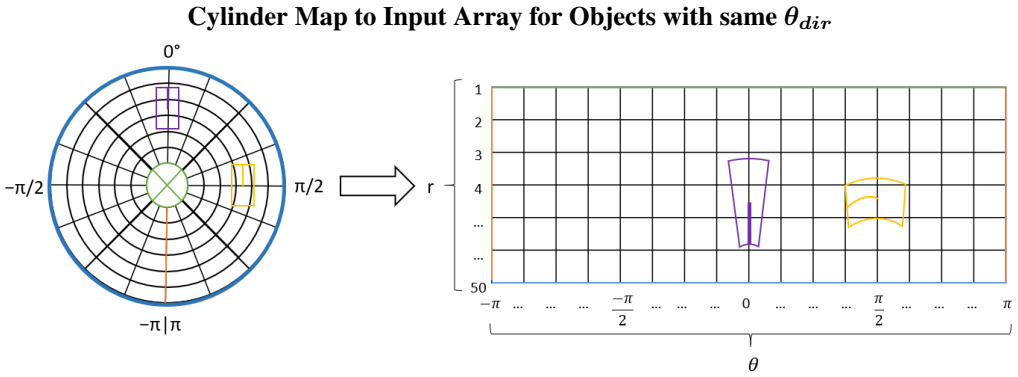

Figure 3. **Illustration of the Cylindrical map translated into the network's 2D array**. Shown here are two identical cars placed at the same distance from the ego vehicle but with different θ_{center}. Since we are working from the sensor's point of view, we can see that it translates *differently* to the 2D array. The network observes the yellow car from the side as the purple car from behind, just as the sensor/driver does. Thus, in order for the network to learn the object's orientation appropriately, a modification is required.

In recent paper RangeRCNN [16] tried to benefit from both range view and BEV. RangeRCNN [16] argued that although range-image is insufficient by itself, incorporating it as a source of initial feature extraction, before transforming the data into BEV architectures can be effective. This method of enhancing BEV with range-image features achieves state-of-the-art results on the KITTI dataset [10].

In our work, we use range information (i.e ρ, the 2D distance in the XY-plane) to guide the 3D convolution layers. In this manner, we gain additional benefit from its data as an initial feature, allowing the network to comprehend objects differently depending on their distance from the ego vehicle. Furthermore, we use range information to manage the receptive field of the voxels and consequently overcome their scale variance in the Cylindrical-coordinate voxels.

3. Challenges in Cylindrical Coordinates

The Cylindrical-coordinate grid differs from the Cartesian in many aspects. First of all, it is a *self-oriented* system. Consequently, identical objects placed at different positions relative to the sensor location will be presented differently along the network's channels. For this reason, in order to adapt the network to the sensor's point of view, certain challenges have to be addressed.

In contrast to Cartesian coordinates BEV, the size of a Cylindrical voxel varies depending on its range. Thus, two identical objects can spread over a different number of voxels, depending on their center location relative to the ego vehicle. Likewise, the orientation of an object toward the ego vehicle might also affect the number of voxels it occupies in the Cylindrical grid. In general, objects spread on a larger amount of voxels, require a wider receptive field in order to detect them.

The Cartesian and Cylindrical-coordinate systems differ in the way they perceive the orientation of objects. The network's point of view can be equated with the perspective of a passenger in the ego-vehicle, observing two identical cars, one of which is in front of the vehicle while the other is on its side. As illustrated in Figure 3, both yellow and purple cars, are of the same size, facing the same way, and at the same range. However, the difference in their θ-centers causes them to be mapped differently to the network's input arrays with an emphasis on the direction. As shown, the purple and yellow cars are no longer facing the same direction in the 2D input array of our network. In order for the network to learn its orientation accordingly, it should be modified in a proper manner.

The same principle applies to the velocity's direction. For example, an object moving along a certain axis will seem to the Cartesian-based network to be moving along that axis regardless of its location on the map. This behavior manifests because the axes of Cartesian coordinates are aligned to those of every other object on the map. However, for the Cylindrical coordinates, this is not the case, and therefore additional adjustments are necessary.

Additionally, since the θ axis is inherently cyclical ($-\pi$ is also π), objects may spread simultaneously over voxels at both the beginning and the end of the network's input arrays. This is a further deviation from the Cartesian system that needs to be taken into account in order to accomplish the transition between the two systems.

In conclusion, transferring from a Cartesian coordinate system to a cylindrical one reveals our networks' unique challenges. To take full advantage of cylindrical coordinates, we must modify the architecture and its learned outputs in order to overcome the aforementioned challenges.

4. Method

After our preprocess of the data our network consists of three learning blocks: (1) Range-Guided Backbone, which

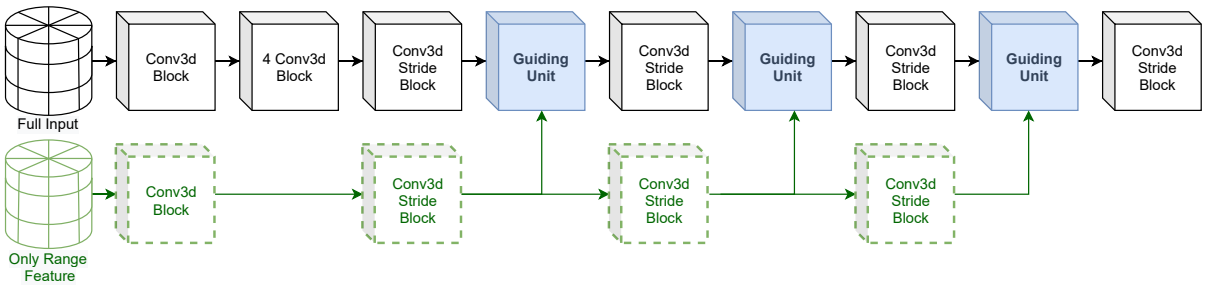

Figure 4. **Illustration of the Range-Guided Backbone:** The *guided* backbone is shown in black, the range *guiding* backbone is shown in a green dashed line and the guiding units are shown in blue.

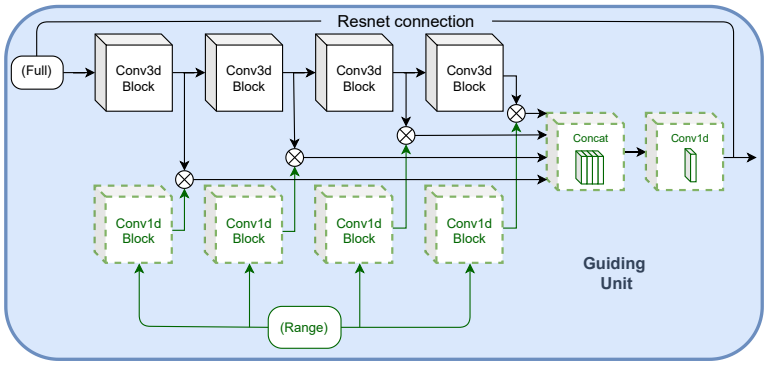

Figure 5. **Illustration of the Guiding Unit:** On the range input we use four separate 1D convolutions. The output of each one is multiplied by the output of its parallel convolution layer on the full input and then concatenated together. In order to achieve the original number of features, we apply 1D convolution with a residual [12] connection.

receives the 3D voxelized input and constructs an output of a flattened 2D map by utilizing our novel guiding units, (2) off-the-shelf Region Proposal Network and (3) a Multi-Group Center Head that outputs a heat-map per class and a variety of predicted parameters for each detected object. Our code is based on the old open source version of CenterPoint [28].

Input: Although the raw output of a LiDAR sensor is given in range and two angles, the nuScenes dataset provides the point cloud sweeps already transformed to Cartesian coordinates. Each point is assigned with two additional features: intensity, which is taken directly from the sensor, and Δt, which relates to the lag-time between each sweep to its key-frame sweep. We followed the rules of the nuScenes detection benchmark by taking ten sequential sweeps (totaling 0.5 seconds). We then quantize it according to the cylindrical voxel resolution and average on each voxel across all features.

Range-Guided Backbone: The range-guided backbone is the first deep network block of our proposed solution. As similar objects appear across a different number of voxels dependent on the range, we propose to control the convolutions in the main pipeline using range-guided mechanism.

The motivation for range guidance is two-fold; First, the LiDAR transmitter releases rapid pulses of laser light across a pre-defined spatial pattern. Using a synced sensor, it measures the time it takes for each pulse to reflect back from the object it hits. Then, we receive multiple hits per object, which depends on its distance. Meaning, nearby objects will be over-represented in contrast to far away objects, as can be seen in Figure 2 leading to significant deviation in close and distant objects statistics. The second reason relates to the voxel size. As noted earlier (see Section 3), in Cylindrical coordinates the voxel size increases with range. Which means nearby objects need a wider receptive field than far away objects.

Our main *guided* backbone is built from four 3D convolution with stride 2 and three guiding units in between as can be seen in Figure 4. The range *guiding* backbone is made of a convolution pipeline fed by only range information (meaning only the range feature of each voxel), controlling the weights multiplying each of the consecutive convolutions output running on the main pipeline.

Specifically, the range backbone is parallel to the main pipeline. In each guiding unit, the output of each 3D convolution running on the main pipeline is multiplied by the

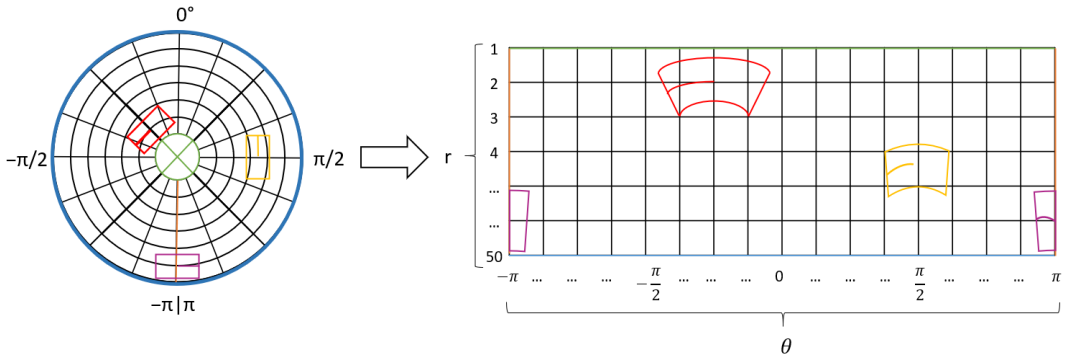

Figure 6. **Illustration of the Cylindrical map translated into the network's 2D array.** Here shown three identical cars of the same size, but with different centers. They have different θ_{dir} as well, but share the same $\bar{\theta}_{dir}$ (defined in equation (3)), the orientation as viewed from the sensor's perspective. In addition, we see that the closer the vehicle is to the sensor, the more voxels it occupies in the θ axis. The circular nature of the θ axis can also be observed here, where the magenta car is spread over both sides of the 2D input array.

output of a 1 x 1 convolution which directly operates on the range feature. The product of four such multiplications are concatenated and then reduced in dimensions by a 1 x 1 convolution operator. See Figure 5 for visual aid.

The design of the guiding unit allows the neural network to learn different receptive fields, while the 3D convolution layers in between contribute to its learning capabilities of the spatial information of neighboring voxels. In our Cylindrical network, this backbone will serve mainly the far range where the receptive field needs to be smaller.

Multi Group Center Head: For this block, we follow the head architecture of CenterPoint [28]. However, in order to overcome the rest of the aforementioned challenges when using cylindrical coordinates, it is necessary to make further modifications for this block.

The original head outputs a K-channel 2D heatmap to indicate the centers of the K-class objects in the x-y plane. In addition to the heatmap, the network outputs ten different scalars for each detection: (dx, dy) which is the delta to the object's center from the heatmap coordinates, (w, l, h) which indicate the object width, length and height, (z_{center}) for the object's center in the Z-axis, (v_x, v_y) for the object's velocity in X and Y axes respectively and $(\cos\theta_{dir}, \sin\theta_{dir})$ where θ_{dir} is the heading angle of the object.

The ground truth heatmaps for training are built by rendering a Gaussian around each object's center. CenterPoint [28] uses symmetric 2D Gaussian since the X and Y axes share the same resolution, and calculate the radius using the following equation:

$$\sigma = \max(f(wl, r), \tau) \tag{1}$$

Where f is the radius function defined in CornerNet [15], τ is the smallest allowable Gaussian radius and w, l are the width and length of the detected object in *number* of voxels, meaning the result of w and l divided by the voxel's size.

In our setup, we can not use symmetric Gaussian anymore thus we split it into 2D Gaussian with different radii for each axis which are calculated separately by eq.(1). We then calculate the size of a voxel in the θ-axis by the following formula:

$$V_{\theta size} = 2r_{center} \sin\left(\frac{\theta_{center}}{2}\right) \tag{2}$$

Where r_{center} , and θ_{center} are the coordinates of the object's center in the r and θ dimensions respectively.

While the transformation of the heatmap coordinates and the deltas from Cartesian to Cylindrical coordinates is quite simple, the heading angle and velocity outputs are much more complex and require a deep understanding of the network's view.

The θ_{dir} in which our Cylinder-network observes the object depends not only on the θ_{dir} of the object but also on the θ_{center} of the object. An object facing forward with $\theta_{center} = \pi/2$ will be interpreted on the 2d map the same way as an object whose center is right in front of the ego-vehicle and facing $\theta_{dir} = -\pi/2$ will, see Figure 6. Therefore, the θ_{dir} we wish to learn is as follows:

$$\bar{\theta}_{dir} = \theta_{dir} - \theta_{center} \tag{3}$$

Where θ_{center} is the θ coordinate of the object's center.

Our use of $\bar{\theta}_{dir}$ instead of θ_{dir} gives us a significant advantage over the Cartesian coordinate system. As demonstrated in Figure 7, points are distributed differently between similar cars with respect to their θ_{center}. Although the two cars share identical θ_{dir} in the classic system, the points are scattered on different parts of each vehicle. As

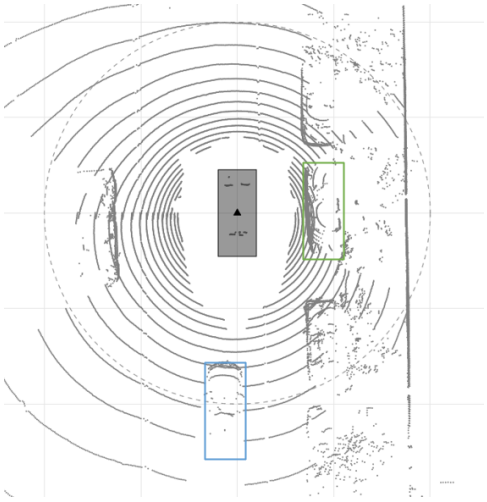

Figure 7. **A single frame of nuScenes scene, demonstrating the Cartesian system's drawback:** Although the green and blue cars share many characteristics, in particular that they both have a similar orientation - the pattern in which they are covered with point clouds is significantly different. By applying the modifications described in equation 3, this difference no longer poses a challenge to our Cylindrical network.

the blue bound car is mostly "hit" on its front, the points "hitting" the green marked car mainly concentrate on its left side. In our modified system, for instance, when most of the object's cloud points cover its rear, the value of its $\bar{\theta}_{dir} = 0$, when the majority of the points appear on its right side, $\bar{\theta}_{dir} = \pi/2$, and so forth. We believe it allows our system a more adequate comprehension of the scene.

Now we can separate the new $\bar{\theta}_{dir}$ into two orthogonal components:

$$\begin{aligned} \theta_\theta &= \sin(\bar{\theta}_{dir}) \\ \theta_r &= \cos(\bar{\theta}_{dir}) \end{aligned} \tag{4}$$

For each detection, one is also required to output the object's velocity. As opposed to θ, the required velocity for this benchmark is already divided into the classical coordinates, meaning the metrics of this benchmark evaluates v_x and v_y and they are given as ground truth. Thus, an additional step is required:

$$\begin{aligned} V_{abs} &= \sqrt{v_x^2 + v_y^2} \\ V_{dir} &= \arctan\left(\frac{v_x}{v_y}\right) \end{aligned} \tag{5}$$

Where we use V_x as the numerator and V_y as the denominator to be aligned with nuScenes coordinate system.
We then apply the same transformation we did in equation (3) for velocity:

$$\bar{\theta}_{velocity} = V_{dir} - \theta_{center} \tag{6}$$

Then, we divide that into two learnable outputs according to our self-cylindrical coordinates as before:

$$\begin{aligned} V_\theta &= V_{abs}\sin(\bar{\theta}_{velocity}) \\ V_r &= V_{abs}\cos(\bar{\theta}_{velocity}). \end{aligned} \tag{7}$$

Losses: We use L_1 regression loss for the local offset $(dr, d\theta)$, orientation $(\theta_r, \theta_\theta)$ as defined in equation (4) and velocity (V_r, V_θ) as defined in equation (7). As for size (w, l, h) and z_{center} we use log-space L_1 regression, and for the heatmap supervision a focal loss [17] with $\alpha = 2$ and $\beta = 4$ is used.

The overall loss of our network is as follows:

$$L = L_{heatmap} + \lambda_{reg} L_{reg} \tag{8}$$

Where $L_{heatmap}$ is the focal loss for the heatmap and L_{reg} and λ_{reg} are the regression losses mentioned above and their loss weight accordingly.

Circular Padding: We address the circularity of the θ axis of the neural network by substituting zero-padding with circular padding for the θ-axis in all of the convolution layers throughout the entire network, both 2D and 3D.

By applying all of the above, we have overcome the challenges posed by the cylindrical coordinate network and can now capitalize on its advantages.

5. Experiments

Dataset: We evaluate our method on the nuScenes dataset for 3D object detection [3]. The dataset contains 700 training scenes, 150 validation scenes, and 150 test scenes, each 20 seconds long with spinning 32 lanes LiDAR at 20 frames per second. The latest nuScenes benchmark requires the detection of 10 different classes with a long-tail distribution: car, bus, construction vehicle, trailer, truck, pedestrian, motorcycle, bicycle, traffic cone and barrier.

Evaluation Metrics: The nuScenes Detection Score (NDS) is a weighted sum of mean Average Precision (mAP) [9] and several True Positive (TP) metrics. The mAP is calculated as an average of four mAP with distance threshold of: (0.5m, 1m, 2m and 4m), and the TP metrics are calculated between the prediction and its *matched* ground truth the under the *2m* threshold:

1. Average Translation Error (**ATE**): The Euclidean distance in meters between the two centers.
2. Average Scale Error (**ASE**): Calculated as (1 - IOU) after aligning the centers and orientation of both.
3. Average Orientation Error (**AOE**): Smallest yaw angle difference (θ_{dir}).

Method	*mAP*↑	*mATE*↓	*mASE*↓	*mAOE*↓	*mAVE*↓	*mAAE*↓	**NDS**↑
CBGS [32]	0.499	0.335	0.256	0.323	**0.251**	0.197	0.613
CenterPoint [28]	**0.591**	**0.277**	**0.251**	**0.269**	<u>0.258</u>	<u>0.189</u>	**0.671**
CyliNet RG (ours)	<u>0.576</u>	<u>0.283</u>	<u>0.253</u>	<u>0.291</u>	0.268	**0.180**	<u>0.661</u>

Table 1. **nuScenes Experiment Results on Validation set:** The results of CBGS and CenterPoint as shown on their github. Top results are in bold, second place is underlined.

Method	*Car*	*Truck*	*Bus*	*Trailer*	*CV*	*Ped*	*Motor*	*Bicycle*	**All**
CenterPoint [28]	3.676	2.637	1.468	14.364	9.197	0.642	**2.641**	**3.155**	**3.051**
CyliNet RG (ours)	**2.610**	**1.485**	**1.273**	**12.964**	**7.267**	**0.561**	3.333	3.974	2.160
Improvement by %:	28.982	43.696	13.291	9.743	20.985	12.535	-26.207	-13.067	**29.223**

Table 2. **Flip Orientation Error on Validation set:** Percentage per class for "flip orientation" error (i.e predicted orientation for an object is nearly 180° from the GT orientation). CV stands for Construction Vehicle, Ped for pedestrian and Motor for Motorcycle.

4. Average Velocity Error (**AVE**): Absolute velocity error for v_x and v_y separately, measured in meters per second.
5. Average Attribute Error (**AAE**): Calculated as 1 - acc, where acc is the attribute classification accuracy.

The final NDS is calculated as follows:

$$NDS = 5mAP + \sum_{i=1}^{5} \max(1 - TP_i, 0.0)$$

NuScenes has recently added a new hidden evaluation metric to their evaluation server, called Planning KL-Divergence [20] (PKL). The PKL metric measures perception performance by measuring differences between how a planner would plan when given detections from a detector instead of human-labeled detections. A larger PKL score corresponds to poorer performance. On the test set, this metric will be displayed.

Implementations Details: Following the nuScenes benchmark, we set the detection range to $[1m, 53.8m]$ for the range axis, $[-\pi, \pi]$ for the θ axis and $[-5m, 3m]$ for the Z-axis. Utilizing the structure of the cylindrical coordinates, we do not need extra spaces in our input map, in order to detect an object within the required 50m range.

With the choice of $[0.075m, \pi/600, 0.2m]$ for the voxels size in each axis respectively, our network input size is 704 x 1200 x 40 voxels. In our range-guided backbone we use four times stride 2 for the range axis, and three times for the θ axis, resulting in 88 x 300 output size. Due to the cylindrical shape of our output map, it is almost 20% smaller than CenterPoint's [28] output map. We train the model with a batch size of 56 for 20 epochs on 8 RTX8000 GPUs. We have chosen adamW [18] optimizer with one-cycle policy [11], LR max 0.0035 with division factor 10, momentum from 0.85 to 0.95, weight decay 0.01 and chose λ_{reg} to be 0.25. During inference, we keep the top 500 detection proposals per sample. Afterward, we filter out all proposals with a detection score lower than 0.1 and apply maximum suppression (NMS) with an IOU threshold of 0.1 for all classes except for the car class where we use 0.2. Ultimately, we limit the maximum output to 83 proposals per group.

5.1. Results

We present our results on both nuScenes validation set shown in Table 1 and nuScenes test set shown in Table 3. The test results were obtained from nuScenes official evaluation server, where test sample annotations remain hidden. Our single model on the LiDAR track outperforms all methods but a single one, both for NDS and PKL scores. Our network also outperforms both of CVCNet's [22] submissions by a very large margin. Although they used the polar coordinate system to quantize the input points cloud, and not the cylindrical one as we did, they are still the only other method that uses another coordinate system than the Cartesian, and therefore we believe a comparison is worthwhile. In NDS we achieved a score of 0.661 versus 0.642 and 0.644 while in the PKL metric (lower is better) we achieved a score of 0.708 as opposed to 0.935 and 0.919 (around 30% improvement).

Moreover, we examined the mAOE and we noticed a fundamental improvement in flip orientation error reduction. Due to the structure of the roads, there are high picks in the orientation distribution for $(0, \pm\pi/2, \pm\pi)$, especially for vehicles. This gives the Cartesian coordinate system an advantage, especially for avoiding small errors in orientation detection, but with the price of increase flip error. Our network on the other hand is comparable on average measuring rotation error, but due to the nature of Cylindrical coordinates point of view we better identify the direction of each element, which is a critical unit for the planners. With the modification we conduct for orientation (eq.3) our

Method	*mAP*↑	*mATE*↓	*mASE*↓	*mAOE*↓	*mAVE*↓	*mAAE*↓	**NDS**↑	**PKL**↓
CBGS [32]	0.528	0.300	0.247	<u>0.379</u>	<u>0.245</u>	0.140	0.633	0.766
CVCNet single [22]	0.558	0.300	0.248	0.431	0.269	**0.119**	0.642	0.935
CVCNet single V2 [22]	0.553	0.300	0.244	0.389	0.268	<u>0.122</u>	0.644	0.919
MMDetection3D [29]	0.575	0.316	0.256	0.409	**0.236**	0.124	0.653	0.710
HotSpotNet-0.1m [6]	<u>0.593</u>	0.274	**0.239**	0.384	0.333	0.133	0.660	0.851
CenterPoint single[28]	**0.603**	**0.262**	**0.239**	**0.361**	0.288	0.136	**0.673**	**0.690**
CyliNet RG single (ours)	0.585	<u>0.272</u>	<u>0.243</u>	0.383	0.293	0.126	<u>0.661</u>	<u>0.708</u>

Table 3. **nuScenes Experiment Results on Test set:** Top results for *single model* on the LiDAR track, taken from the nuScenes website. Top results are in bold, second place is underlined.

CC.mod.	*0.5m*↑	*1m*↑	*2m*↑	*4m*↑	*mAP*↑	*mATE*↓	*mASE*↓	*mAOE*↓	*mAVE*↓	*mAAE*↓	*NDS*↑
×	0.468	0.561	0.618	0.648	0.574	**0.282**	**0.250**	0.314	0.312	0.185	0.652
√	**0.469**	**0.562**	**0.623**	**0.651**	**0.576**	0.283	0.253	**0.291**	**0.268**	**0.180**	**0.661**

Table 4. **Ablation Experiment on nuScenes Validation set:** Cylindrical-coordinate modifications (CC.mod.) refers to modification explained on subsection Multi Group Center Head under section Method 4 for both orientation ($\bar{\theta}_{dir}$) and velocity ($\bar{\theta}_{velocity}$). Notice, how our CC.mod. significantly improves performance for both, orientation (mAOE) and velocity (mAVE).

Cylindrical network is learning the relative orientation as shown and explained in Figures 6 and 7, therefore when learning for example $\bar{\theta}_{dir} = 0$ it is always related the same orientation in space, in this case a rear view of the object. As Table 2 indicates, this modification led to a decrease in "flip orientation" error. In this table we show the percentage per class for such error, averaging on all four of nuScenes mAP metrics. As can be seen, our network is decreasing the error rate in almost all classes by a large margin, and improves results by 29.2% when combining all classes in all mAP together.

5.2. Ablation studies

We further conduct our ablation studies on the nuScenes validation set.

First, we demonstrate the significance of our Cylindrical-coordinate modifications detailed in Multi-Group Center Head on section 4. Table 4 shows the result for two identical networks, trained in the same way, both utilizing circular padding. In the first network, we did not use our velocity and orientation modifications, while in the second, we did. As expected, the most significant improvement in the second network's performance can be observed in the velocity and orientation metrics. The mean average orientation error decreases from 0.314 to 0.291 and the mean average velocity error decreases from 0.312 to 0.268 (a 15% improvement). Additionally, all the mAP metrics indicate an increase as well, and we see a significant increase in the NDS from 0.652 to 0.661.

Moreover, we show the effectiveness of our range-guided backbone to reduce the receptive field for larger distances and for smaller objects. For range $[30m, 50m]$ over all classes we achieve NDS of 0.485 when the guidance mechanism is applied, and 0.478 when omitted while using the heavily optimized backbone of CBGS [32]. For the entire range we also witness an improvement in NDS from 0.658 when omitted versus 0.661 when applied. An interesting point to note is that the range information is included in our Cylindrical network even without the guiding units. Nevertheless, we see here that an additional use of it as convolution weight guidance in the first block, is proving beneficial for the network.

6. Conclusions

In this paper, we explore the vast potential of a Cylindrical-coordinate based network for representing LiDAR point clouds in an outdoor scene. We present the first end-to-end framework for 3D object detection in those coordinates along with a novel range-guided backbone that controls the receptive field based on the range information. We further report a fundamental improvement in flip orientation error which is critical to the planner in identifying potential risks. Our method achieves strong results on the highly challenging nuScenes dataset and lays the foundation for further Cylindrical methods in this domain.

7. Acknowledgment

This work is partially funded by the Zimin Institute for Engineering Solutions Advancing Better Lives, the Israeli consortiums for soft robotics and autonomous driving, the Nicholas and Elizabeth Slezak Super Center for Cardiac Research and Biomedical Engineering at Tel Aviv University and TAU Science Data and AI Center.

References

[1] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*, 2019.

[2] Alex Bewley, Pei Sun, Thomas Mensink, Dragomir Anguelov, and Cristian Sminchisescu. Range conditioned dilated convolutions for scale invariant 3d object detection, 2020.

[3] 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. *arXiv preprint arXiv:1903.11027*, 2019.

[4] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, 2017.

[5] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2018.

[6] Qi Chen, Lin Sun, Zhixin Wang, Kui Jia, and Alan Yuille. Object as hotspots: An anchor-free 3d object detection approach via firing of hotspots, 2020.

[7] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[8] Yilun Chen, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Fast point r-cnn. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[9] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vision*, 88(2):303–338, June 2010.

[10] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[11] Sylvain Gugger. The 1cycle policy. https://sgugger.github.io/the-1cycle-policy.html.

[12] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[13] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Lake Steven Waslander. Joint 3d proposal generation and object detection from view aggregation. *Intelligent Robots and Systems (IROS)*, 2018.

[14] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019.

[15] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[16] Zhidong Liang, Ming Zhang, Zehan Zhang, Xian Zhao, and Shiliang Pu. Rangercnn: Towards fast and accurate 3d object detection with range image representation, 2020.

[17] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007, 2017.

[18] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[19] Gregory P. Meyer, Ankit Laddha, Eric Kee, Carlos Vallespi-Gonzalez, and Carl K. Wellington. Lasernet: An efficient probabilistic 3d object detector for autonomous driving. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[20] Jonah Philion, Amlan Kar, and Sanja Fidler. Learning to evaluate perception models using planner-centric metrics. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2020.

[21] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 5099–5108. Curran Associates, Inc., 2017.

[22] Ernest Cheung Qi Chen, Lin Sun and Alan L. Yuille. Every view counts: Cross-view consistency in 3d object detection with hybrid-cylindrical-spherical voxelization. In *Advances in Neural Information Processing Systems 33 preproceedings (NeurIPS)*, 2020.

[23] 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. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[24] 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. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP:1–1, 02 2020.

[25] P. Wang, P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou, and G. Cottrell. Understanding convolution for semantic segmentation. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1451–1460, 2018.

[26] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, Oct 2018.

[27] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[28] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3d object detection and tracking, 2020.

[29] Wenwei Zhang, Kai Chen, Zhe Wang, Jianping Shi, and Chen Change Loy. Mmdetection3d. https://github.com/open-mmlab/mmdetection3d.

[30] Yang Zhang, Zixiang Zhou, Philip David, Xiangyu Yue, Zerong Xi, Boqing Gong, and Hassan Foroosh. Polarnet: An improved grid representation for online lidar point clouds semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[31] Y. Zhou and O. Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018.

[32] Benjin Zhu, Zhengkai Jiang, Xiangxin Zhou, Zeming Li, and Gang Yu. Class-balanced grouping and sampling for point cloud 3d object detection, 2019.