Fully Convolutional One-Stage 3D Object Detection on LiDAR Range Images

Anonymous Author(s)
Affiliation
Address
email

Abstract

We present a simple yet effective fully convolutional one-stage 3D object detector for LiDAR point clouds of autonomous driving scenes, termed FCOS-LiDAR. Unlike the dominant methods that use the bird-eye view (BEV), our proposed detector detects objects from the range view (RV, *a.k.a.* range image) of the LiDAR points. Due to the range view's compactness and compatibility with the LiDAR sensors' sampling process on self-driving cars, the range view-based object detector can be realized by solely exploiting the vanilla 2D convolutions, departing from the BEV-based methods which often involve complicated voxelization operations and sparse convolutions.

For the first time, we show that an RV-based 3D detector with standard 2D convolutions alone can achieve comparable performance to state-of-the-art BEV-based detectors while being significantly faster and simpler. More importantly, almost all previous range view-based detectors only focus on single-frame point clouds since it is challenging to fuse multi-frame point clouds into a single range view. In this work, we tackle this challenging issue with a novel range view projection mechanism, and for the first time demonstrate the benefits of fusing multi-frame point clouds for a range-view based detector. Extensive experiments on nuScenes show the superiority of our proposed method and we believe that our work can be strong evidence that an RV-based 3D detector can compare favourably with the current mainstream BEV-based detectors.

1 Introduction

With the rise of autonomous driving, 3D object detection from the LiDAR point cloud has been recently drawing increasing attention. Similar to the 2D image object detection [1, 2, 3, 4, 5], 3D object detection requires the model to predict the (3D) locations of the objects of interest and the associated properties (*e.g.*, categories, sizes, heading, and the state of motion). In spite of the unprecedented success that the computer vision community has attained on the 2D image object detection, it is still intractable to transfer the success to the 3D object detection task.

Most previous 3D object detection methods consider that the point cloud is amorphous and consists of a set of unordered points. Thus, this task is considered significantly different from its 2D detection counterpart, which works on structured RGB images. Moreover, the cornerstones of modern computer vision – CNNs or CNN-like vision transformers (*e.g.*, Swin Transformers [6]) also assume that the inputs are well-organized as grids, which poses another difficulty of accurate 3D object detection in point clouds. As a result, in order to adopt these well-developed techniques, almost all top-performing point cloud object detection methods first partition the 3D space into *structured* voxels (or pillars) [7, 8, 9, 10, 11] and then follow a paradigm similar to that of 2D object detectors [1, 12]. Additionally, given the prior that it is very rare two objects being stacked along with the elevation

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

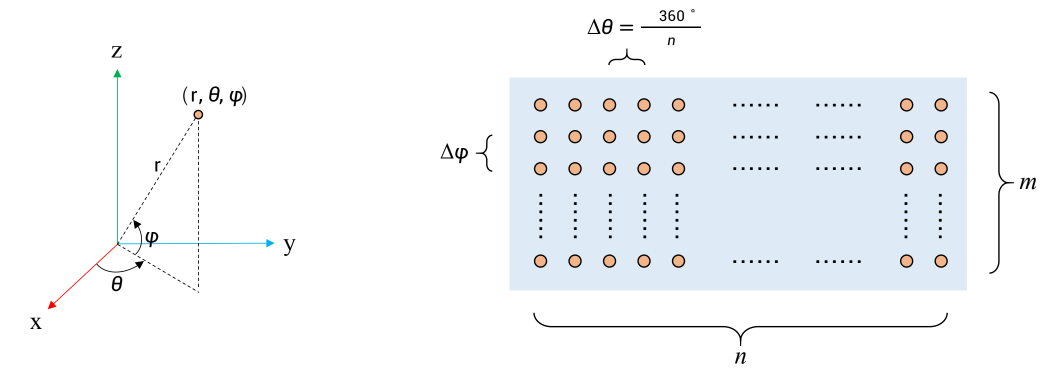

Figure 1: **The range image of the point cloud (right).** The size of the range image is $m \times n$, where m is the number of beams and n is the number of measurements (*i.e.*, sampling frequency) per scan cycle. A 3D point's coordinates on the range image are computed by discretizing the azimuthal angle θ and the inclination angle ϕ in the spherical coordinate system (left). The m, n, and $\Delta\phi$ depend on the LiDAR specifications.

axis in autonomous driving scenes, most methods only carry out the detection task on the bird-eye view (BEV) of the point cloud, which can reduce the exponentially increasing complexity resulted from the third dimension. However, owing to incompatibility with the LiDAR's sampling process in the autonomous driving scenes, BEV-based solutions often suffer from the following shortcomings. 1) In fact, the points in autonomous driving are regularly sampled in the spherical coordinate system with the origin being the LiDAR sensor. The BEV disregards this regularity, and causes the issue that the voxels far away from the origin have much fewer points than the ones near the origin; and a large number of voxels are even empty. This results in the need for sparse convolutions [9, 8, 13], significantly complicating the system, particularly for on-device applications. 2) The points in a voxel have to be sampled or padded so that every voxel has the same number of points. The sampling decimates a large number of points, leading to the loss of information before the model see anything. 3) From the BEV, some objects such as "pedestrian" and "traffic cone" become very small. Accurately detecting these objects requires a fine-grained voxel size, dramatically increasing the price of computation.

In this work, we advocate a new solution for 3D object detection that works on the range view (RV). As noted by many previous works [14, 15, 16], if we consider these points in the spherical coordinate system and project them in terms of their inclination and azimuth angles, they can form a compact 2D image with size being $m \times n$ (shown in Fig. 1), where m is the number of beams (*i.e.*, channels) of the LiDAR sensor and n is the sampling frequency per scan cycle. The resulting image is referred to as "range image" or "range view" of the point cloud. Compared to the aforementioned BEV, the range image is nearly dense and compatible with the LiDAR sampling process, eliminating the need for sparse convolutions and alleviating the loss of points. In addition, the range image closely resembles the common RGB image, minimizing the cost of transferring the 2D detection methods to 3D ones. In the literature, some works attempted to detect objects on the range view such as RangeDet [14] and LaserNet [16]. These works have shown that RV-based methods can also achieve decent detection performance, showing the promise. However, previous RV-based methods only focus on the single-frame point cloud as the aforementioned range view structure does not hold anymore if the ego (and the origin) moves between the multiple frames. Additionally, the range image of a single-frame point cloud is already nearly dense so that there are not many vacancies the points from other frames can populate. These issues make the range-view based detectors difficult to benefit from the multi-frame fusion. In sharp contrast, the multi-frame fusion can dramatically improve the performance in the BEV-based detectors, as shown in [8, 10]. This makes the performance of RV-based detectors largely lag behind that of the BEV-based ones, hampering their development and application. In this work, we show this issue can be largely remedied with a well-designed Multi-round Range View (MRV) projection mechanism. The proposed MRV makes the RV-based detectors be able to enjoy the gain of multi-frame fusion and thus achieve competitive performance with multi-frame BEV-based detectors.

Here, we summarize our main contributions as follows.

- We propose a fully convolutional one-stage 3D object detector, termed FCOS-LiDAR. FCOS-LiDAR works on the LiDAR range images and minimizes the gap between the 3D and 2D detectors, while being substantially simpler than current mainstream BEV-based 3D detectors [8, 7].
- Compared to previous BEV-based detectors [7, 8], FCOS-LiDAR sidesteps the complicated voxelization process and eliminates the need for sparse convolutions due to the highly compact range view representation of the point cloud. In the setting of only using the single-frame point cloud as inputs, FCOS-LiDAR can outperform the state-of-the-art BEV-based detector CenterPoint [8] while being significantly faster.
- We also present a well-designed Multi-round Range View (MRV) projection mechanism, making RV-based detectors be able to benefit from the multi-frame fusion of point clouds as well, and achieve competitive performance compared to the multi-frame BEV-based detectors. To our knowledge, we are the first one approaching the challenge of the RV-based multi-frame point clouds fusion and showing that RV-based detectors can also be boosted by the multi-frame fusion of point clouds.
- We believe that our excellent performance of the RV-based detector can be a strong evidence that RV-based detectors compare favorably against the mainstream BEV-based detectors and encourage the community to pay attention to this promising solution.

2 Related Work

Bird-view based 3D Detection. Most top-performing LiDAR-based 3D detectors [17, 9, 8, 10, 7] fall into this category, which first convert the point cloud into BEV images. VoxelNet [7] is the first end-to-end BEV-based detector, which employs PointNet [18] to handle the representation within a voxel and 3D convolutions to generate high-level features for the region proposal network (RPN). SECOND [9] proposes to use sparse convolutions, which can save the computing burden of 3D convolutions. Another popular approach is to eliminate the voxelization along the elevation axis and convert the point cloud into the pillars [10]. Based on the voxel-based or pillar-based BEV representation, CenterPoint [8] achieves state-of-the-art performance by using the anchor-free pipelines.

Range-view based 3D Detection. Due to the compactness of the RV representation, some methods [19, 20, 16, 16] also attempt to perform detection based on the representation. VeloFCN [21] is the pioneering work to perform 3D objection using the range view, which transforms point cloud to the range image and then applies 2D convolution to detect 3D objects. After that, some following works [16, 14] are proposed to narrow the performance gap between RV-based and BEV-based detectors. LaserNet [16] models the distribution of 3D box corners to capture their uncertainty, resulting in more accurate defections. RCD [20] introduces the range-conditioned dilation mechanism to dynamically adjust the dilation rate in terms of the measured range, which can alleviate the scale-sensitivity issue of the RV-based detectors. RangeDet [14] further proposes the Range Conditioned Pyramid to mitigate the scale-variation issue and utilizes the Meta-Kernel convolution to better exploit the 3D geometric information of the points. To our knowledge, these existing RV-based detectors only take into consideration the single frame point cloud and neglect the substantial improvements brought by the multi-frame fusion as shown in BEV-based detectors.

3 Our Approach

3.1 Range View Representation

Given a LiDAR point (x, y, z) in the Cartesian coordinate system with the z axis pointing upward, it can be uniquely transformed to the spherical coordinates (r, θ, ϕ) with

$$r = \sqrt{x^2 + y^2 + z^2},\ \theta = \mathrm{atan2}(y, x),\ \phi = \mathrm{atan2}(z, \sqrt{x^2 + y^2}), \tag{1}$$

where r, θ, and ϕ are the range, azimuthal angle, and inclination angle, respectively, as shown in Fig. 1(left). The LiDAR samples the points with a fixed number of beams (denoted by m), each of which has a fixed inclination angle. These LiDAR beams synchronously rotate around the z axis uniformly to obtain a 360° horizontal field of view and the LiDAR measures a certain number of times (denoted by n) per scan cycle (*i.e.*, per frame). Thus, the difference of adjacent measurements' azimuthal angles is ${}^{360}/_{n}{}^{\circ}$. For example, on the nuScenes dataset, the LiDAR measures $n = 1086$ times per scan cycle and has $m = 32$ beams. The inclination angles of these beams are evenly spaced from -30.67° to 10.67°, inclusive. Note that the inclination angles are not always evenly spaced and subject to the specifications of the LiDAR.

Given the regularity of the azimuthal and inclination angles for a given LiDAR, we can discretize the azimuthal angles with n bins, and inclination angles with m bins, respectively. Let (i, j) be the indices of the bins for azimuthal and inclination angles, respectively. By computing all the pairs of (i, j) of the points in a single scan cycle, we can fill these points into a 2D image $I \in \mathbb{R}^{m \times n \times C}$ (*i.e.*, the range image), where $C = 9$ consists of the original Cartesian coordinates (x, y, z), the spherical coordinates (r, θ, ϕ), the reflected intensity i, the existence e of the point, and a relative timestamp t. The existence denotes whether or not the location is filled by a point, and the relative timestamp is only valid in the multi-frame point cloud inputs and denotes the time difference between the frame containing this point and the current frame. In practice, the vehicle itself is often in motion, and this causes that some points might be projected to the same bins on the range image. In this case, we keep the one with the minimal distance to the vehicle.

3.2 Multi-round Range View Projection (MRV)

The point cloud of a single frame is often sparse in the 3D space and of low resolutions. In order to improve the detection performance, the current frame is often combined with several previous frames as the network's inputs [8, 7]. Taking the nuScenes dataset as an example, the methods on this dataset often take as the inputs 10 frames, which is composed of the current frame and previous 9 frames, including ∼240K points in total. The crucial issue of the range view representation is the collision that multiple points fall into the same bin happens much more frequently in the multi-frame case. For instance, on nuScenes, only ∼28K points are finally kept in the range image and ∼90% of the points are discarded due to the collision. The decimation makes the multi-frame point cloud have almost the same number of valid points with the single-frame version, which is the dominant reason that RV-based methods cannot enjoy the benefit of multi-frame inputs.

In order to cope with the crucial issue, we propose the Multi-round Range View (MRV) projection mechanism. To be specific, we first project the points with the method in Sec. 3.1. Then, instead of discarding the rejected points, we project them again with the same process and put the them in another group of nine channels. This projection process is repeated until a sufficient number of points are kept. On the nuScenes dataset, this is repeated five times and the percentage of the kept points can be improved from ∼10% to more than 50%. In theory, as long as we continue the process, all of the points can be kept. However, we found that the performance is saturated after 5 repetitions on the nuScenes dataset. Finally, the resulting range images of the five rounds are concatenated along the channel dimension and used as the inputs. Additionally, one caveat is that the points of the current frame should have the highest priority wherever the collision happens because the points of previous frames are stale and might not reflect the current status of the world. Despite being a very simple treatment, it significantly affects the effectiveness of the multi-frame fusion as shown in our experiments.

3.3 Modality-wise Convolutions

As mentioned before, each pixel on the single-frame range image contains nine channels. Different the RGB image, whose three channels are of the same modality (*i.e.*, in the color space) and correlated for the manifold of natural images, the nine channels of the range image are not like that and belong to five modalities, *i.e.*, $[x, y, z]$, $[r, \theta, \phi]$, $[i]$, $[e]$ and $[t]$, respectively, where the channels in the one pair of square brackets are of the same modality. For example, the correlation between the reflected intensity i and the coordinate x does not make sense. Further, even the different channel types in the same modality are orthogonal and less correlated as well. For instance, it is difficult to say there is a relationship between the azimuthal angle θ and the range r of a point. As a result, one channel type should be viewed as an individual "modality". By default, the conv. layer simultaneously compute spatial correlations and cross-channel correlations. However, as shown before, the channels of the range image are less correlated, and thus, it is not reasonable to use the default conv. layer here, and the channels of the range image should be processed separately. This can be easily implemented with the grouped convolutions. Here, we term it *modality-wise convolution* because it is based on the

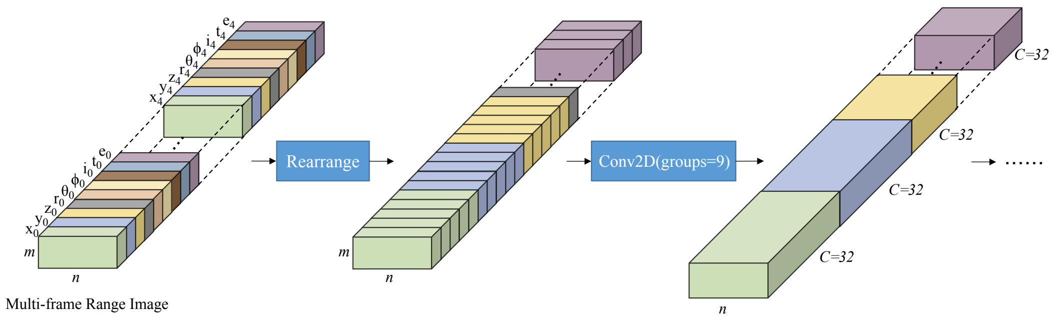

Figure 2: **Modality-wise convolutions with a multi-frame range image.** As we can see, we first rearrange the channels of the multi-frame range image so that the channels with the same type from different frames (*e.g.*, x_0, x_1, ... x_{T-1}) are adjacent. Then, two successive 2D conv. layers with the number of groups being 9 (which is equal to the number of different channel types) are used to process these channels separately, mapping each channel type to a 32-channel features. Finally, the features of these channel types are merged by a 1×1 conv. layer.

"modalities". Once the high-level semantic features of these modalities are individually obtained, we can aggregate the features of these modalities with a 1×1 conv. layer (*i.e.*, point-wise convolution) for further abstract analysis.

In this way, the modality-wise convolution is analogous to the widely-used depth-wise convolution, but we highlight that the underlying nature is distinct. Our modality-wise convolution instantiates a reasonable *inductive bias* based on the prior that different modalities are less relevant, and thus it is expected to simultaneously improve both the effectiveness and efficiency, as shown in our experiments. This also leads to the fact that the modality-wise convolution can only be placed at the beginning of the network, where the channels are interpretable and have an explicit modality. In contrast, the depth-wise convolution (together with the point-wise convolution) is often viewed as an efficient approximation of the full conv. layer and thus it does not usually yield improved performance and can appear anywhere. Lastly, when it comes to the multi-frame point clouds, the channels of the same type from different frames should be handled together because they are closely correlated. The whole procedure of the modality-wise convolution is illustrated in Fig. 2.

3.4 Overall Architecture

The overall architecture of FCOS-LiDAR is shown in Fig. 3. FCOS-LiDAR follows the spirit of the anchor-free detector FCOS [2] in the image-based object detection and is a standard fully convolutional network [22].

Taking a (multi-frame) range image $I \in \mathbb{R}^{m \times n \times (T \times C)}$ as an example, where T is the number of the used frames, we first forward the range image through our backbone network, termed LiDAR-Net. LiDAR-Net is adapted from ResNet-50 [23]. Specifically, before the first conv. layer of ResNet-50, we insert three branches of the modality-wise convolutions with three dilation rates being 1, 3, and 6, respectively. The branches with various dilation rates aim to capture the multi-scale context, whose outputs are summed up. Then, the first two $2\times$ downsamplings of ResNet-50 are removed, which is of great importance due to the low resolutions of the range images. Moreover, we change the numbers of blocks of ResNet-50's four stages from $(3, 4, 6, 3)$ to $(4, 4, 1, 1)$ and stop doubling the number of channels in the third and fourth stages because we found that the final performance is not sensitive to the capacity of the later stages but quite sensitive to that of the early stages. This ravels one of the important difference between the LiDAR-based and image-based object detection tasks. For object detection in RGB images, more convolutions in the later stages are often required to transform the RGB pixels into the highly semantic and abstract features that can be used to obtain the geometries of the objects. However, the LiDAR points themselves are already geometric points and thus that many convolutions in the later stages are no longer needed. Thus, we can instead allocate the capacity in the later stages to the early stages (before any downsampling) to better incorporate the geometric information carried by the raw points. Finally, the four levels of feature maps from LiDAR-Net's four stages are used, denoted by C_2, C_3, C_4, and C_5, respectively.

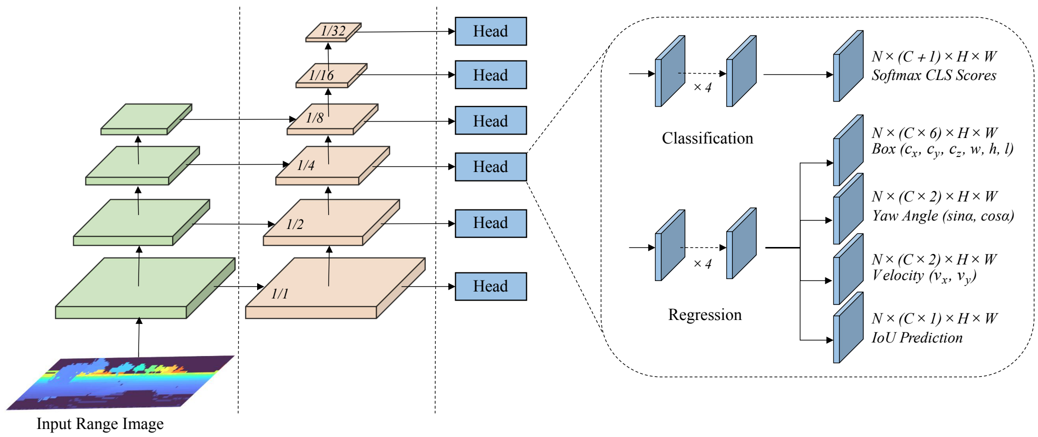

Figure 3: **Overall architecture.** The overall architecture of FCOS-LiDAR resembles the 2D image-based detector FCOS [2]. By taking as input the range image, the network obtains the multi-level FPN features, and then the classification and regression branches are attached to these feature levels to predict the final 3D boxes. Different from FCOS, the weights of the detection heads are not shared between the FPN levels as mentioned in Sec. 3.4. In addition, the class-specific regression heads are used instead of the class-agnostic ones in FCOS.

Next, following FCOS, the four levels of feature maps are sent into a feature pyramid network (FPN) [24] to obtain six levels of pyramid feature maps denoted by P_2, P_3, P_4, P_5, P_6, and P_7. Their spatial downsampling ratios to the input are $1/1$, $1/2$, $1/4$, $1/8$, $1/16$, and $1/32$, respectively. Then, similar to FCOS, the classification and regression heads, each with four 3×3 conv. layers of channel 64 and a final prediction conv. layer (for the classification or regression), are attached to these feature levels. Unlike the heads in image-based detectors, whose weights are shared between these feature levels, it is important in the LiDAR-based detector to untie these weights. This is another important difference between image-based and LiDAR-based detectors. In image-based detectors, different sizes of objects can be normalized to similar sizes of objects by the downsampling the image with corresponding factors. That is what the "pyramid" means in the FPN. Thus, the image-based detectors can share the weights of the detection heads because the objects have been normalized to similar sizes after the FPN. However, in LiDAR-based detection, the objects' sizes cannot be normalized in this way because the sizes of the objects are determined by the 3D points' coordinate values in them and downsampling the range images cannot alter their real sizes in the 3D space. Therefore, it is on longer reasonable to share the detection heads between these FPN levels. This is also confirmed in our experiments.

Training Targets Computation. Similar to FCOS [2], we need to assign the training targets to each pixel on the range image. The training targets computation is described using the nuScenes dataset as an example. On nuScenes, an object's ground-truth 3D box is parameterized by $(c_x^*, c_y^*, c_z^*, w^*, h^*, l^*, \alpha^*)$, where (c_x^*, c_y^*, c_z^*) is the 3D center of the box, and w, h, l, and α respectively are the width, height, length, and the yaw angle around the z axis. If the original 3D point of a pixel on the range image is contained in the 3D box of an object, the pixel is responsible for the object and predicts its category, 3D box and etc.. Here, the 3D box regression targets of the pixel are relative to the original 3D coordinates of the range image pixel and are defined as

$$\Delta c_x = c_x^* - x_0,\ \Delta c_y = c_y^* - y_0,\ \Delta c_z = c_z^* - z_0,\ t_w = \log(w^*),\ t_h = \log(h^*),\ t_l = \log(l^*), \quad (2)$$

where (x_0, y_0, z_0) are the original 3D coordinates of the range image pixel. Similarly, the regression targets of the yaw angle α are also relative to the azimuthal angle of the pixel, and following the convention [8], we decouple the relative azimuthal angle into $(\sin \Delta\alpha, \cos \Delta\alpha)$. On nuScenes, we also need to predict the velocity vector (v_x^*, v_y^*) of the object, which are used as the training targets as is. Other pixels whose 3D points are not in any 3D box are used as the negative samples. Note that the points within a object's 3D box form a 2D mask on the range image.

Network Outputs. The nuScenes dataset has $C = 10$ classes of interest, which is predicted by the classification head followed by a $\mathrm{softmax}$ layer (the upper head in Fig. 3). The other regression targets are predicted by four sibling output heads of the regression branch, respectively, as shown in Fig. 3. Here, we use the class-specific regression predictions and thus the number of the output channels is amplified by C times. Moreover, following [25, 5], each positive pixel also predicts the intersection-over-union (IoU) between the predicted 3D box and ground-truth one, which is multiplied to the classification score before the non-maximum suppression (NMS).

Loss Functions. The classification predictions are supervised with the cross entropy (CE) loss. Following [26], we dynamically reassign the classification labels during training. Specifically, for a ground-truth 3D box, only the top K pixels whose predicted 3D boxes have the lowest costs with the ground-truth box are assigned with the positive labels and other pixels are considered negative, where the cost is defined as the summation of the classification loss and the opposite of the IoU between the predicted boxes and the ground-truth box, and K is dynamically calculated, being the summation (rounded to an integer) of the highest $Q = 20$ IoUs between the predicted boxes and the ground-truth box. For the 3D box regression, we make use of both the IoU loss [27] and L_1 loss. Following [2], the IoU predictions are penalized with the binary cross entropy (BCE) loss since they are in the range $[0, 1]$.

Inference. The inference is very similar to that of the 2D image-based detector FCOS. To be specific, the range images are forwarded through the network and the aforementioned predictions are obtained. The predictions are filtered in terms of the classification scores and only the predictions with the score greater than 0.01 are kept. The 3D boxes are restored by inverting the computation of the training targets. Then, the NMS on the 3D boxes is applied with threshold 0.2 to remove the duplications. Finally, the top 500 predictions with the highest scores are used as the final predictions.

4 Experiments

We conduct experiments on the nuScenes dataset [28], which contains 1000 scenes with 700, 150, and 150 scenes for training, validation, and testing, respectively. The training, validation, and testing sets have 28K, 6K, 6K keyframes annotated with 10 classes, respectively. For all ablation experiments, we train the models on the training set and report the performance on the validation set unless specified. The metrics of the 3D detection task are mean Average Precision (mAP) and the nuScenes detection score (NDS). The LiDAR sensor of the nuScenes dataset has $m = 32$ beams and $n = 1086$ measurements per scan cycle. The mAP is based the center distances on the bird-eye view at thresholds 0.5m, 1m, 2m, 4m in place of the box IoUs. NDS is a weighted average of mAP, the translation error, the scale error, the orientation error, the velocity error, and the box attributes error.

Implementation Details. Unless specified, following [8], FCOS-LiDAR is trained by 40 epochs with the AdamW [29] optimizer under the MMDetection3D framework [30], which takes ∼26 hours on 8 A100 GPUs. The one-cycle learning rate policy [31] with initial learning rate 10^{-3} is used. The learning rate gradually increases to 0.01 in the first 40% epochs and then gradually decreases to 10^{-7} in the rest of the training process. The weight decay is 0.01, and the momentum ranges from 0.85 to 0.95. In addition, due to the low vertical resolution of the range image on the nuScenes dataset, we upscale the range image in the vertical direction by 2 with the nearest interpolation. During training, the point cloud is randomly flipped along both x and y axes and rotated in the range $[-\pi, \pi]$, as well as globally scaled by a random factor from [0.95, 1.05]. The ground-truth copy-paste data augmentation from [9] is also used. For multi-frame point cloud, we use 10 sweeps in total, as in previous works [8, 10]. The inference time is measured on a 3090Ti GPU with batch size 1.

4.1 Multi-round Range View Projection

Here, we conduct experiments to demonstrate the effectiveness of the proposed multi-round range view (MRV) projection. As shown in Table 1, in the single-frame settings, MRV is also helpful, for example, by increasing the number of rounds from 1 to 3, the mAP can be boosted by 0.62%. This is due to the fact that the vehicle is often in motion and the aforementioned collisions happen within a single frame as well. As you can see, if one round is used, the multi-frame fusion can improve the mAP by a substantial margin ∼1.9% (from 53.14% to 55.02%). The improvement can be dramatically increased to 3.9% mAP if we use 5 rounds in MRV (57.8% mAP), due to the fact that 5-round MRV can keep much more points in the multi-frame point clouds as mentioned in Sec. 3.2. This confirmed the effectiveness of MRV. Note that elapsed time of MRV is insensitive to the number of rounds as shown in Table 1.

Table 1: **Multi-round range view (MRV) projection.** Time: the elapsed time of MRV.

		Single-frame		Multi-frame	
#Rounds	Time (ms)	mAP (%)	NDS (%)	mAP (%)	NDS (%)
1	**0.66**	53.14	51.52	55.02	61.27
3	0.72	**53.76**	**53.62**	56.01	62.82
5	0.76	53.42	53.40	**57.08**	63.15
7	0.76	53.06	53.10	56.99	**63.47**

Table 2: **Whether to make the points of the current frame have the highest priority.** "C.V.", "Ped." and "C.T." indicate "construction vehicle", "pedestrian" and "traffic cone", respectively.

Prioritized?	mAP(%)	NDS(%)	Car	Truck	Bus	Trailer	C.V.	Ped.	Motor	Bicycle	T.C.	Barrier
	54.29	62.31	76.8	47.3	62.0	32.0	16.5	80.3	53.8	**38.2**	70.1	65.9
✓	**57.08**	**63.15**	**82.1**	**52.3**	**65.2**	**33.6**	**18.3**	**84.1**	**58.5**	35.3	**73.4**	**67.9**

More importantly, in the RV-based multi-frame settings, it is crucial to assign the highest priority to the points from the current frame when the collision happens. As shown in Table 2, without this treatment, the performance of the multi-frame fusion is significantly dropped from 57.08% to 54.29% in mAP. Note that the single-frame counterpart can already achieve mAP 53.42%. We argue that the neglect of this point in previous works is one of the main reasons that RV-based detectors can barely enjoy the benefit of multi-frame fusion.

4.2 Modality-wise Convolutions

Table 3: **Modality-wise convolutions.** "Multi-frame": whether to group together the channels with the same type from multiple frames. Time: the latency of this module.

Modality groups	Multi-frame	Time (ms)	mAP (%)	NDS (%)
$[x, y, z, r, \theta, \phi, i, t, e]$	✓	6.0	56.35	62.65
$[x, y, z]$, $[r, \theta, \phi]$, $[i]$, $[t]$, $[e]$	✓	4.2	56.34	63.07
$[x]$, $[y]$, $[z]$, $[r]$, $[\theta]$, $[\phi]$, $[i]$, $[t]$, $[e]$	✓	**3.2**	**57.08**	**63.15**
$[x]$, $[y]$, $[z]$, $[r]$, $[\theta]$, $[\phi]$, $[i]$, $[t]$, $[e]$		3.8	56.83	**63.15**

The experimental results of our modality-wise convolutions are shown in Table 3. If we use the full convolutions here, which compute the correlation between all the channels of the range image, FCOS-LiDAR can achieve 56.35% in mAP. By splitting the channels into various modalities (2nd row), the performance can be slightly improved, *i.e.*, from NDS 62.65% to 63.07%. Moreover, due to the fact that even the different channel types of the same modality are orthogonal and less correlated, we process each channel type individually. As shown in the table, the performance can be further boosted from mAP 56.34% to 57.08% (3rd row) while the lowest latency is achieved. Finally, the last row shows the results if we do not consider the channels with the same type from multiple frames together, where the mAP is slightly worse.

In addition, as mentioned before, we employ multiple modality-wise convolution branches with various dilation rates in parallel to capture the multi-scale context. The experiments are shown in Table 4. As you can see, compared with the one using only one dilation rate, using multiple dilation rates can improve the performance by more than 1% in mAP (from 55.87% to 57.08%).

4.3 Untied Weights of Detection Heads

As mentioned before, it is better to untie the weights of the detection heads between the FPN levels in the LiDAR-based detector. This is confirmed in Table 5. As we can see, by untying the weights, the performance can be improved from mAP 56.44% to 57.08%. This ravels one of the interesting differences between image-based and LiDAR-based detectors because the shared detection heads between FPN levels often achieve better performance in image-based detectors [2, 24].

4.4 Inference Time Comparisons

We compare the inference time of FCOS-LiDAR and the state-of-the-art BEV-based detector CenterPoint [8]. As shown in Table 6, the proposed MRV is faster than the voxelization in CenterPoint. It is worth noting that the voxelization algorithm reported here is *nondeterministic*, which cannot yield stable results but being significantly faster. In the official code of MMDetection3D [30], the deterministic voxelization takes $\sim$100ms for the multi-frame point clouds. In sharp contrast, the

Table 4: **Multi-scale context aggregation.**

Dilations	mAP(%)	NDS(%)	Car	Truck	Bus	Trailer	C.V.	Ped.	Motor	Bicycle	T.C.	Barrier
$(1,)$	55.87	62.68	81.8	51.0	64.1	32.6	16.9	83.8	56.3	31.2	72.7	**68.3**
$(1,3)$	56.39	62.96	**82.3**	52.1	**65.3**	32.8	17.1	83.7	58.2	32.5	72.0	67.8
$(1,3,6)$	**57.08**	**63.15**	82.1	**52.3**	65.2	**33.6**	**18.3**	**84.1**	58.5	**35.3**	**73.4**	67.9
$(1,1,1)$	56.63	63.12	82.1	51.5	65.2	32.9	17.1	83.7	**58.8**	33.7	73.1	68.0

Table 5: **Whether to untie the weights of the detection heads.**

Untied?	mAP(%)	NDS(%)	Car	Truck	Bus	Trailer	C.V.	Ped.	Motor	Bicycle	T.C.	Barrier
	56.44	63.09	82.0	51.1	64.3	31.1	**18.3**	83.8	57.9	34.7	72.9	**68.4**
✓	**57.08**	**63.15**	**82.1**	**52.3**	**65.2**	**33.6**	**18.3**	**84.1**	**58.5**	**35.3**	**73.4**	67.9

Table 6: **Inference time breakdowns.** We compare against the state-of-the-art BEV-based CenterPoint [8], which trained with exactly the same strategies. FCOS-LiDAR is significantly faster as well as competitive in the multi-frame setting (and superior in the single-frame setting).

Method	Voxel/MRV(ms)	Backbone(ms)	Heads(ms)	Overall(ms)	mAP(%)	NDS(%)
single-frame point cloud						
CenterPoint [8]	1.62	41	8	50.62	52.83	**53.86**
FCOS-LiDAR (**Ours**)	**0.63**	**31**	**7**	**38.63**	**53.42**	53.40
multi-frame point cloud						
CenterPoint [8]	1.95	63	9	73.95	**60.40**	**67.25**
FCOS-LiDAR (**Ours**)	**0.76**	**31**	**7**	**38.76**	57.08	63.15

proposed MRV is always *deterministic* and highly efficient. Moreover, due to the compactness of the range image, our network can be implemented with the standard convolutions alone, thus being much more efficient than CenterPoint using the sparse convolutions as shown in the table. The elapsed time of the post-processing is omitted here as it is closely similar in both methods.

4.5 Comparisons with State-of-the-art Methods

We further compare FCOS-LiDAR with other state-of-the-art methods on the nuScenes test set. For the model on the test set, we increase the number of channels in the detection heads from 64 to 128, which improve the validation mAP by ∼0.6% with slightly longer latency. Additionally, we remove the copy-paste data augmentation in the last 5 epochs during training as in [32]. As shown in Table 7, FCOS-LiDAR achieves competitive performance with other state-of-the-art methods.

Table 7: **Comparisons with state-of-the-art methods on the nuScenes test set.** The results are directly taken from their original papers.

Method	mAP(%)	NDS(%)	Car	Truck	Bus	Trailer	C.V.	Ped.	Motor	Bicycle	T.C.	Barrier
PointPillars [10]	30.5	45.3	68.4	23.0	28.2	23.4	4.1	59.7	27.4	1.1	30.8	38.9
SSN [33]	46.3	56.9	80.7	37.5	39.9	43.9	14.6	72.3	43.7	20.1	54.2	56.3
CVCNet [34]	55.3	64.4	82.7	46.1	46.6	49.4	22.6	79.8	59.1	31.4	65.6	69.6
CBGS [8]	52.8	63.3	81.1	48.5	54.9	42.9	10.5	80.1	51.5	22.3	70.9	65.7
CenterPoint [8]	58.0	**65.5**	**84.6**	**51.0**	**60.2**	**53.2**	17.5	83.4	53.7	23.7	76.7	**70.9**
FCOS-LiDAR (c128)	**58.8**	64.8	81.7	45.8	52.3	49.0	**27.5**	**83.7**	**64.1**	**35.8**	**77.9**	70.0

5 Conclusion

We have presented an efficient range-view-based 3D object detector FCOS-LiDAR. FCOS-LiDAR shows the challenging LiDAR-based object detection can also be solved with the standard convolutions alone, similar to what we have done in the image-based 2D object detection. We also for the first time show the RV-based 3D detector can also enjoy the benefit of the multi-frame fusion with the proposed MRV. We hope our strong results can encourage the community to pay more attention to this promising direction.

Societal Impacts. This paper presents a method that can locate the 3D location of the objects of interest in LiDAR point cloud. This technique might be abused for military purposes, for example, on lethal autonomous weapons.

References

[1] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Proc. Advances in Neural Inf. Process. Syst.*, pages 91–99, 2015.

[2] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. FCOS: Fully convolutional one-stage object detection. In *Proc. IEEE Int. Conf. Comp. Vis.*, pages 9627–9636, 2019.

[3] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[4] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[5] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[6] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.

[7] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4490–4499, 2018.

[8] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11784–11793, 2021.

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

[10] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019.

[11] Bo Li. 3d fully convolutional network for vehicle detection in point cloud. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1513–1518. IEEE, 2017.

[12] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv: Comp. Res. Repository*, 2019.

[13] Benjamin Graham. Spatially-sparse convolutional neural networks. *arXiv preprint arXiv:1409.6070*, 2014.

[14] Lue Fan, Xuan Xiong, Feng Wang, Naiyan Wang, and Zhaoxiang Zhang. Rangedet: In defense of range view for lidar-based 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2918–2927, 2021.

[15] 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. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020.

[16] 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12677–12686, 2019.

[17] Lue Fan, Ziqi Pang, Tianyuan Zhang, Yu-Xiong Wang, Hang Zhao, Feng Wang, Naiyan Wang, and Zhaoxiang Zhang. Embracing single stride 3d object detector with sparse transformer. *arXiv preprint arXiv:2112.06375*, 2021.

[18] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[19] Pei Sun, Weiyue Wang, Yuning Chai, Gamaleldin Elsayed, Alex Bewley, Xiao Zhang, Cristian Sminchisescu, and Dragomir Anguelov. Rsn: Range sparse net for efficient, accurate lidar 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5725–5734, 2021.

[20] Alex Bewley, Pei Sun, Thomas Mensink, Dragomir Anguelov, and Cristian Sminchisescu. Range conditioned dilated convolutions for scale invariant 3d object detection. In *Conference on Robot Learning*, pages 627–641. PMLR, 2021.

[21] Bo Li, Tianlei Zhang, and Tian Xia. Vehicle detection from 3d lidar using fully convolutional network. *arXiv preprint arXiv:1608.07916*, 2016.

[22] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 3431–3440, 2015.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[24] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 2117–2125, 2017.

[25] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*, 2021.

[26] Zheng Ge, Songtao Liu, Zeming Li, Osamu Yoshie, and Jian Sun. Ota: Optimal transport assignment for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 303–312, 2021.

[27] Jiahui Yu, Yuning Jiang, Zhangyang Wang, Zhimin Cao, and Thomas Huang. Unitbox: An advanced object detection network. In *Proceedings of the 24th ACM international conference on Multimedia*, pages 516–520, 2016.

[28] 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. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[29] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[30] MMDetection3D Contributors. MMDetection3D: OpenMMLab next-generation platform for general 3D object detection. `https://github.com/open-mmlab/mmdetection3d`, 2020.

[31] Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, volume 11006, page 1100612. International Society for Optics and Photonics, 2019.

[32] Chunwei Wang, Chao Ma, Ming Zhu, and Xiaokang Yang. Pointaugmenting: Cross-modal augmentation for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11794–11803, 2021.

[33] Xinge Zhu, Yuexin Ma, Tai Wang, Yan Xu, Jianping Shi, and Dahua Lin. Ssn: Shape signature networks for multi-class object detection from point clouds. In *European Conference on Computer Vision*, pages 581–597. Springer, 2020.

[34] Qi Chen, Lin Sun, Ernest Cheung, and Alan L Yuille. Every view counts: Cross-view consistency in 3d object detection with hybrid-cylindrical-spherical voxelization. *Advances in Neural Information Processing Systems*, 33:21224–21235, 2020.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] In set multi-frame settings, the RV-based detector can still not outperform the BEV-based ones in mAP despite being significantly faster.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See the conclusion section.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] This work is in progress now, and we will release the code upon the acceptance.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] The used dataset is quite large and none of previous works reports the error bars on this dataset.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See implementation details.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]