Towards Efficient LiDAR-Based Perception in Autonomous Driving: Exploiting Point Cloud Sub-sampling and Camera-based Prior Knowledge

Markus Schön*, Michael Kösel*, Klaus Dietmayer, Michael Buchholz

Abstract: Cameras and LiDAR are among the most commonly used sensors for autonomous driving applications; often, a combination of both is used. Although both sensors have attracted considerable interest, camera-based approaches are the most mature, thanks partly to industry efforts to optimize the underlying processing pipelines. In contrast, LiDAR-based perception methods still rely on the research community's efforts to develop and support the processing libraries that make the methods possible. This imbalance between the perception methods of the two sensor modalities often results in a high GPU computational load and long inference times for the processing of LiDAR point clouds, which may be too demanding for the embedded devices in autonomous vehicles. In this work, we explore point cloud sub-sampling using prior knowledge to reduce the computational burden of LiDAR processing. In particular, we show that with perfect prior knowledge, the accuracy of modern 3D object detectors is not affected even when large portions of point clouds are removed. For practical applications, we instead use 2D object detections of surrounding cameras to determine relevant regions. In extensive experiments, we demonstrate that our proposed framework can reduce the computational load and inference time of 3D object detectors while maintaining high detection performances.

Keywords: Camera-based Prior Knowledge, Efficient Perception, LiDAR 3D Object Detection, Point Cloud Sub-sampling

1 Introduction

Autonomous vehicles use multiple sensors to perceive the environment reliably. Among the most common sensor combinations are camera and LiDAR sensors. Camera sensors provide rich semantic information about the environment, including lane markings and traffic signs. On the other hand, LiDAR sensors allow for accurate depth measurements at long ranges independent of the lighting conditions.

In recent years, both sensors have seen a significant improvement in perception methods, mainly thanks to the development of deep learning approaches. In particular, camera sensors, a well-established technology, have benefited from the efforts of both the research

*Equal contribution

All authors are with the Institute of Measurement, Control, and Microtechnology of the University of Ulm, 89081, Ulm, Germany. E-Mail: {firstname}.{lastname}@uni-ulm.de

Part of this work is accomplished within the project AUTOtech.*agil* (FKZ 01IS22088W). We acknowledge the financial support for the project by the Federal Ministry of Education and Research of Germany (BMBF).

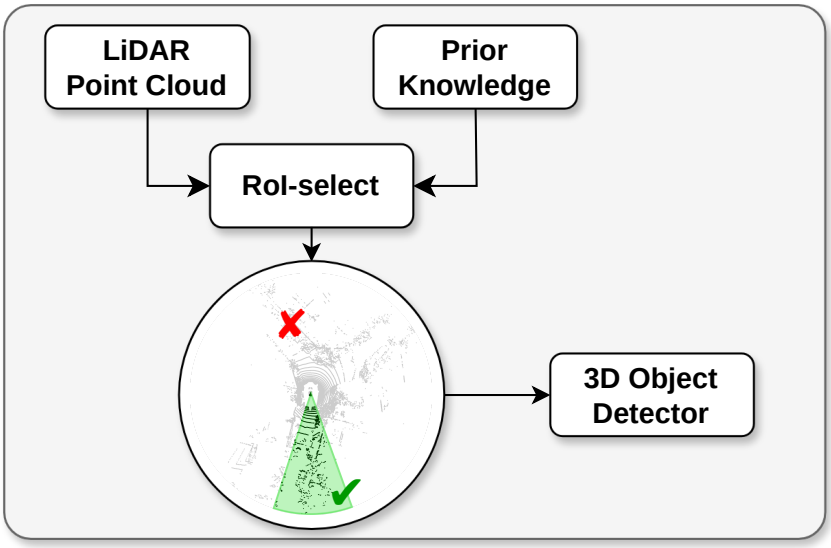

Figure 1: Overview of our proposed framework. Given prior information, we can remove irrelevant sectors from a LiDAR point cloud, thus decreasing the memory consumption and inference time.

community and industry to improve the performance and reliability of these methods. For example, methods such as YOLOv8 [1] allow high object detection performance with low computational and inference costs. In contrast, LiDAR sensors, which have only recently become a commonly used sensor, still lag in both computational and speed performance, with methods often requiring large amounts of GPU memory and inference time to achieve high performance. This imbalance between camera and LiDAR performance is mainly because 3D point processing frameworks are still in their infancy and are primarily community-developed and maintained (e.g., SpConv [2], TorchSparse [3]), as opposed to being an industry standard (e.g., PyTorch [4], JAX [5]). As autonomous driving features are introduced into production vehicles, the hardware required to run such methods is often reduced to embedded devices. Therefore, the high computational cost that LiDAR sensors can introduce into the processing pipeline may be a deciding factor in their widespread adoption.

To address the current computation cost of LiDAR sensors, we propose a simple yet effective framework for reducing the computational load of 3D point clouds by using prior information about the surrounding environment. Fig. 1 gives an overview of our proposed framework. In particular, given a multi-sensor setup that includes both camera and LiDAR sensors, we investigate how information extracted from a computationally efficient source (e.g., 2D bounding boxes extracted from camera images) can be used to select regions of the 3D point cloud where relevant objects are present. Our extensive experiments show that the accuracy of popular 3D object detectors is unaffected if the prior information used is correct, even when large portions of the LiDAR point cloud are discarded, while drastically reducing both computational load and inference time. In real applications, where prior information might not be perfect, our experiments on the popular nuScenes dataset show that using the YOLOv8 object detector in surrounding cameras can lead to comparable object detection performances while improving computation times.

In summary, our main contributions are:

- We investigate the use of region dropping in LiDAR point clouds to improve the computational load and inference time of popular 3D object detectors.
- We explore different region-dropping strategies, including sector-dropping and beam down-sampling.
- In extensive experiments on the nuScenes dataset, we show that in the ideal case of perfect prior knowledge, the performance of several 3D object detectors can be perfectly maintained while reducing inference time up to 40%.
- Additionally, for real-world applications, we show that using fast and efficient 2D object detectors such as YOLOv8 can lead to comparable performance to using full point cloud processing while reducing computation load and inference times.

2 Related Work

This section reviews existing work on image-based object detection, LiDAR-based 3D object detection, and methods that integrate prior knowledge to improve runtime performance.

2.1 Image-based Object Detection

Camera-based perception algorithms have seen significant advantages in recent years, mainly driven by the rapid development of deep learning methods, cheaper computation hardware, and larger and richer datasets. Among the most popular computer vision perception tasks is 2D object detection. One of the seminal works in this domain is R-CNN [6], which introduced the idea of region proposal networks to localize relevant objects prior to classification. Follow-up works, such as Fast R-CNN [7] and Faster R-CNN [8], further contribute to the region proposal idea by improving both speed and accuracy. In contrast, single-stage approaches, such as SSD [9] and YOLO [10], avoid the expensive region proposal stage, allowing even faster inference times. YOLO has been adopted by many researchers and seen continual iterations of improvement over the years, with YOLOv8 [1] being one of the latest incarnations, offering astonishing speed and performance on several benchmarks. The high speed of modern image-based object detectors can mainly be addressed to the use of efficient 2D convolutions to extract relevant features. Modern deep learning frameworks, like PyTorch [4], TensorFlow [11], and JAX [5], all support 2D convolutions, allowing to leverage the ever-increasing hardware capabilities of GPUs for faster training and inference. In recent years, transformer-based architectures have also been explored under the family of vision-transformers (ViTs) [12]. Although transformer-based architectures have shown great potential in many different tasks, their inference performances have only recently reached comparable results with CNN-based methods.

2.2 LiDAR-based 3D Object Detection

Compared to camera images, which are in a structured format, 3D point clouds are unstructured and order invariant, making the feature extraction process more complicated. Early works in 3D object detection tackled this problem by projecting the point cloud in a structured voxel grid. Detectors, such as VoxelNet [13], leveraged this structure to perform full 3D convolutions on the point cloud. SECOND [14] builds on the idea of VoxelNet but uses sparse convolutions to improve training and inference speed significantly. Although showing good performance, the voxel formulation has the disadvantage that most of the voxels are empty due to the sparsity of LiDAR point clouds (often reaching 99.99% of empty voxels [15]). Consequently, most of the computation is wasted on empty voxels. Other methods of structured projections have been explored in the literature to improve efficiency. PointPillars [16] projects the point cloud in a pillar grid (a 2D grid with a single height) and then uses 2D convolutions to extract features. BEVDetNet [17] uses a bird's eye view (BEV) projection instead, which, unlike the pillar grids, does not use height features during processing. On the other hand, range-based methods, such as [18], project the point cloud into a 2D image using a spherical coordinate system and then treat it as an image to extract features. Recently, sparse 3D convolutions have been used for 3D point cloud processing, often resulting in high-performing detectors such as CenterPoint [19]. Compared to traditional convolutions, sparse operations avoid computation in empty voxels, drastically reducing computation times. Although libraries such as SpConv [2] and TorchSparse [3] have allowed for such methods to be developed, sparse convolutions are still not part of the core industry standard frameworks. Compared to 2D image-based convolutions, current 3D sparse convolution methods still lag behind in terms of computational load and inference time.

2.3 3D Object Detection using Prior Knowledge

The usage of prior knowledge to improve the efficiency of 3D object detectors has been investigated before. Henning *et al.* [20] propose to reduce the number of processed LiDAR point clouds by introducing an event-based frame-dropping based on 2D object detections. While this approach can save computations, the performance decreases with an increased number of dropped frames. In contrast, our method can save computations while maintaining detection performance. Frustum-based LiDAR processing introduced by Qi *et al.* [21] adds camera information to the LiDAR processing pipeline. By projecting a 3D frustum of each camera detection into the point cloud, they can select areas, which are then processed by a PointNet [22]. Frustum ConvNet [23] extends the frustum-based approach by using convolutional neural networks (CNNs) to capture local spatial features better. Paigwar *et al.* [24] propose Frustum-PointPillars, which combines the benefits of both [21, 23] by using pillars [16], which further improves the performance. Our method is most similar to frustum-based approaches. However, instead of using a single camera view to generate frustums based on each camera detection, our method uses multi-view surround cameras and uses the 2D detections as priors to drop whole sectors of the LiDAR point cloud.

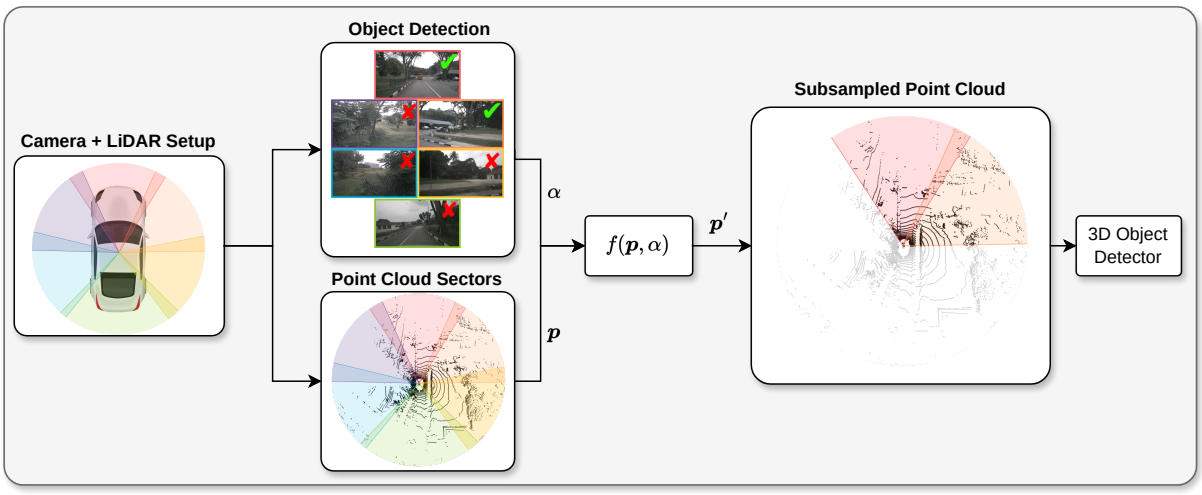

Figure 2: Overview of our method. We assume a surround-view camera setup combined with a 360° LiDAR sensor. Then, we split the point cloud $\mathbf{p}$ into N_S sectors, where each sector corresponds to the field of view of a camera. Now, object detection is performed for each camera and each point cloud sector, where the corresponding camera did not detect any object is dropped. Finally, using the prior-knowledge α extracted from the 2D object detection, the subsampled point cloud $\mathbf{p}'$ is derived using the RoI-select function $f(\mathbf{p}, \alpha)$ used as input to a 3D object detector, leading to improved inference times and less memory consumption.

3 Method

This section presents the proposed framework, outlining both reprocessing with ground truth prior knowledge and the adaptation to real-world applications using surrounding multi-camera images. An overview of our proposed method is given in Fig. 2.

3.1 Problem Setting

We assume that an autonomous vehicle is equipped with a 360° LiDAR sensor and surrounding view multi-camera setup. Assuming both sensors are synchronized and well-calibrated, at time step t, the LiDAR sensor delivers a point cloud $\mathbf{p} \in \mathbb{R}^{N \times C}$, with N total points, each defined by C features, i.e., x, y, z, intensity. Furthermore, the camera sensors deliver a set of images $I = \{\mathbf{i}_1, \ldots, \mathbf{i}_\ell\}$, where ℓ is the total number of cameras and $\mathbf{i} \in \mathbb{R}^{H \times W \times D}$ is a single image with sizes H, W and D. Assuming that the LiDAR sensor and the camera images have an overlapping field of view, our method aims to identify which regions of the LiDAR point cloud $\mathbf{p}$ contain relevant objects. In particular, we aim to define the region-of-interest selection function (RoI-select) $f(\mathbf{p}, \alpha) : \mathbb{R}^{N \times C} \rightarrow \mathbb{R}^{N' \times C}$, which takes as input the point cloud $\mathbf{p}$ and a set of prior-knowledge information α and returns a point cloud $\mathbf{p}'$ which has N' points, with $N' \leq N$.

The prior information α can be derived from multiple sources, e.g., ground truth information, offline maps, real-time traffic information, or other sensors. In our approach, we rely on two source types: ground truth 3D object detections derived from an oracle source (e.g., label information) and 2D bounding box objects extracted from the camera images. We use the first source (ground-truth information) as a proof of concept to determine the

upper-bounds performances of our approach and the second source (2D bounding boxes from camera images) as a real-world application of our proposed framework.

3.2 Region of Interest Selection via Ground Truth Priors

As input, it is assumed that the LiDAR point cloud $\mathbf{p}$ and the prior knowledge α composed of the set of 3D bounding boxes B are present. Here, the set B represents the relevant objects in the scene, with $B = \{\mathbf{b}_1, \ldots, \mathbf{b}_M\}$, $\mathbf{b} = [x, y, z, w, l, h, \theta]$ expressing the position, size, and orientation of the object and M the total number of boxes. We divide $\mathbf{p}$ into N_S polar sectors $S = \{s_1, \ldots, s_{N_S}\}$ to select the relevant regions in the point cloud. To do this, we first project the (x, y, z) Cartesian coordinates of each point in $\mathbf{p}$ in spherical coordinates (r, ϕ, σ), where $r = \sqrt{x^2 + y^2 + z^2}$, $\sigma = \text{atan2}(\sqrt{x^2 + y^2}, z)$ and $\phi = \text{atan2}(y, x)$. Then, we divide the point cloud into N_S equally spaced sectors S, each containing a subset of points of $\mathbf{p}$. The RoI-select function, which in this case takes as input $f(\mathbf{p}, B)$, selects a sector $s \in S$ if it includes points that are part of one of the objects determined by B. Sectors that do not contain any relevant objects are discarded. The final point cloud $\mathbf{p}'$ is then composed by concatenating all valid sectors s.

3.3 Region of Interest Selection via 2D Image Detection

In real-world applications, ground truth prior information like the ones described in the previous section is often unavailable. Therefore, other sources of prior knowledge need to be derived. As mentioned, we rely on 2D object detections from surrounding camera images to extract such information. In particular, given a set of images I, we perform object detection in each image $\mathbf{i} \in I$ using the fast and computationally efficient YOLOv8 architecture. This operation results in a set of 2D detection boxes D for each camera image, which can be used to define the RoI-select function $f(\mathbf{p}, D)$. As we assume to have a surrounding camera system with a full 360° field of view, each camera can be associated with a specific section of the LiDAR point cloud. Therefore, as in the previous section, we divide $\mathbf{p}$ into N_S different sectors S, where N_S is the total number of cameras ℓ. Unlike before, where a sector was discarded if it did not contain points belonging to a relevant object, here, a sector is kept if there are 2D object detections of relevant classes in the camera image. Afterward, the output point cloud $\mathbf{p}'$ is the concatenation of all the points belonging to the kept sectors. A visual example of the sector selection process is shown in Fig. 2.

3.4 Additional Methods for Region Selection

Thus far, sector dropping has been assumed to reduce the total number of input points. However, the proposed framework allows for different point selection approaches. For example, beam sector down-sampling can be used to reduce the number of points in the vertical field of view of the LiDAR sensor. Given a down-sampling factor $d \in \mathbb{N}$, the number of points can be reduced by simply choosing every d increment of azimuth angle ϕ. More advanced methods can also be used if different prior knowledge is available. For example, if semantic segmentation is performed on the camera images instead of object detection, only masks containing relevant object points could be selected, further reducing the number of input points.

4 Experimental Results

This section extensively evaluates our proposed approach on multiple standard LiDAR object detectors. In addition, we also demonstrate the effect of perfect prior information on inference time and accuracy.

4.1 Dataset

We conduct experiments on the multi-modal automated driving dataset nuScenes [25]. The nuScenes dataset consists of 1000 driving scenes recorded in Boston and Singapore. Each scene is captured using six cameras, a 32-beam LiDAR, and five radar sensors. The LiDAR sensor can return up to 1.4 M points per second [25]. This work only uses the provided 360° camera data and the LiDAR point clouds. To evaluate the object detection performance, we report the standard nuScenes metrics, i.e., the mean average precision (mAP) and the nuScenes detection score (NDS).

4.2 Implementation Details

We implement our method in PyTorch [4] using the OpenPCDet [26] framework. All experiments were performed on an NVIDIA GeForce RTX 3090. Our method is detector-agnostic and is applied only during inference. To validate the effectiveness of our approach, we test our method on the following LiDAR object detectors: PointPillars [16], SECOND [14], and CenterPoint [19]. All detectors were trained using the default settings used by OpenPCDet. Note that the point cloud range is different for different detectors, resulting in a different amount of input points. For the camera object detector, we use the YOLOv8 [1, 27] detector trained on the BDD100K [28] dataset. The YOLOv8 architecture is chosen due to its fast inference time and ability to generalize to other datasets, such as nuScenes.

4.3 Results

We evaluate our method on the nuScenes validation set. Since the nuScenes dataset also provides sweeps of intermediate timesteps, we also report the results of using multiple sweeps as input, leading to an even higher amount of points.

Region of Interest Selection via Ground Truth Priors. In Tables 1 and 2, the results using the ground-truth priors in the RoI-select function are reported. The mean and standard deviation of the inference time were measured over the entire nuScenes validation set. In the case of single sweep input processing, we can see that compared with the entire point cloud input, all three detectors achieve similar mAP and NDS scores. In the case of PointPillars, we see a slight reduction in performance (-1.48 mAP, -0.99 NDS) when using $N_S = 50$, which highlights that, although not fundamental, the surrounding context, which might be dropped from the region selection, can help improve object detection performances. Conversely, we see that in the case of CenterPoint, sector dropping helps improve detection performances by small margins. For example, in the case of $N_S = 10$, the performance improves by $+0.54$ mAP and $+0.27$ NDS. This effect can be mainly explained by reducing false positive detections, avoided by not processing

Table 1: Evaluation results of the proposed method using ground truth prior knowledge (Section 3.2). The experiments are conducted in the nuScenes validation set using a single sweep input for the object detectors. In the table, the performance for the *full* input processing as well as different values of the sector dropping parameter N_S are reported. The detection metrics mAP and NDS are in percentage.

Detector	**Sectors**	**mAP ↑**	**NDS ↑**	**Inference Time (ms)**	**Num Points**	**Num Pred Boxes**
CenterPoint	*full*	49.34	49.79	49.70 ± 2.02	34288	151
	6	49.71	49.97	46.88 ± 1.44	27770	139
	10	49.88	50.06	46.40 ± 1.55	25025	134
	20	49.84	50.02	45.94 ± 1.63	21322	127
	30	49.76	50.00	45.65 ± 1.64	19234	122
	40	49.64	49.94	45.40 ± 1.72	17888	120
	50	49.49	49.84	45.48 ± 1.58	16894	118
PointPillars	*full*	40.95	43.79	27.00 ± 1.38	34195	49
	6	40.89	43.72	25.47 ± 1.34	27688	48
	10	40.84	43.65	25.21 ± 1.34	24949	47
	20	40.40	43.34	25.13 ± 1.37	21257	47
	30	39.96	43.09	25.17 ± 1.34	19176	46
	40	39.69	42.90	25.15 ± 1.33	17834	45
	50	39.47	42.80	25.14 ± 1.34	16843	45
SECOND	*full*	47.38	48.29	36.88 ± 1.64	34195	52
	6	47.42	48.26	34.66 ± 1.82	27688	51
	10	47.55	48.37	34.37 ± 1.92	24949	50
	20	47.41	48.20	33.85 ± 1.98	21257	50
	30	47.21	47.98	33.64 ± 2.12	19176	49
	40	47.24	47.96	33.33 ± 1.94	17834	49
	50	47.07	47.86	33.33 ± 2.10	16843	48

nonrelevant point cloud regions. This is also reflected in the average number of predicted boxes for the whole point cloud (151 boxes) and the reduced input (134 boxes).

When observing the number of input points, we see that a considerable reduction is possible. For example, in the case of CenterPoint and $N_S = 50$, a reduction of 17394 points can be achieved, which is approximately 50 % fewer points. In the case of multi-sweep inputs, which generally lead to a more significant number of input points, a similar trend can be observed in both performance and the number of input points. As the total number of points is usually directly connected to the number of voxels or pillars that a 3D object backbone processes, the observed reduction dramatically reduces the detector's computational burden. This is particularly true for sparse convolution backbones like the one used by SECOND or CenterPoint, where convolutions are performed only on non-empty voxels. Therefore, by appropriately removing regions of space where relevant objects are not present, the computation load of an object detector can be reduced without any architectural changes.

Table 2: Evaluation results of the proposed method using ground truth prior knowledge (Section 3.2). The experiments are conducted in the nuScenes validation set using a multi sweep input (10 sweeps) for the object detectors. In the table, the performance for the *full* input processing as well as different values of the sector dropping parameter N_S are reported. The detection metrics mAP and NDS are in percentage.

Detector	**Sectors**	**mAP ↑**	**NDS ↑**	**Inference Time (ms)**	**Num Points**	**Num Pred Boxes**
CenterPoint	*full*	59.21	66.47	74.61 ± 14.75	256866	132
	6	59.59	66.65	54.13 ± 2.91	200731	123
	10	59.76	66.73	53.24 ± 2.88	179235	119
	20	59.89	66.76	52.51 ± 2.87	152642	113
	30	59.98	66.72	51.96 ± 2.73	137878	109
	40	60.17	66.81	51.64 ± 2.71	129086	107
	50	59.97	66.76	51.19 ± 2.72	122856	106
PointPillars	*full*	44.70	58.29	48.86 ± 14.65	255936	93
	6	44.87	58.42	29.79 ± 1.14	199902	87
	10	44.92	58.37	29.59 ± 1.14	178467	83
	20	44.86	58.29	29.43 ± 1.24	151980	79
	30	44.88	58.27	29.33 ± 1.23	137284	77
	40	44.92	58.26	29.31 ± 1.27	128530	75
	50	44.79	58.21	29.18 ± 1.28	122335	74
SECOND	*full*	50.65	62.39	58.50 ± 14.06	255936	74
	6	51.48	62.85	40.82 ± 2.12	199902	72
	10	51.78	63.00	40.33 ± 2.22	178467	70
	20	52.04	63.15	39.80 ± 2.29	151980	68
	30	52.27	63.24	39.35 ± 2.37	137284	66
	40	52.48	63.33	39.12 ± 2.32	128530	65
	50	52.45	63.33	38.94 ± 2.39	122335	64

In Tables 1 and 2, we also report the inference times measured on the validation set. Although the absolute numbers heavily depend on the GPU hardware used for inference, the relative improvement shows that a substantial reduction in inference time can be achieved. This is especially true for the multi-sweep input, which, as mentioned before, leads to a much higher number of input points. For example, in the case of PointPillars with $N_S = 50$, the inference time is reduced with respect to the total input processing by 19.68 ms approximately which corresponds to roughly 40 % faster inference time. A similar trend is also observed for CenterPoint (31 % faster inference) and SECOND (33 % faster inference time). For the single sweep inputs, there is also an improvement in inference time, albeit to a lesser extent.

Region of Interest Selection via 2D Image Detection. In Table 3, we report the results using prior information from the 2D detections from the six surrounding cameras. Similar to the ground truth prior knowledge usage described above, it can be seen that a reduction in the total amount of input points and inference time can be achieved. For

Table 3: Evaluation results of the proposed method using prior knowledge derived from computing 2D detections in the surrounding camera images (Section 3.3). The experiments are conducted on the nuScenes validation set using both single sweep and multi sweep (10 sweeps) inputs for the object detectors. The detection metrics mAP and NDS are in percentage.

Multi Sweep	Detector	RoI Select	mAP	NDS	Inference Time (ms)	Num Points	Num Pred Boxes
✓	CenterPoint	✗	59.21	66.47	74.61 ± 14.75	256866	132
		✓	57.42	65.42	54.28 ± 2.93	211924	125
	PointPillars	✗	44.70	58.29	48.86 ± 14.65	255936	93
		✓	43.06	57.35	29.44 ± 1.21	211070	88
	SECOND	✗	50.65	62.39	58.50 ± 14.06	255936	74
		✓	49.60	61.84	40.73 ± 2.18	211070	72
✗	CenterPoint	✗	49.34	49.79	49.70 ± 2.02	34288	151
		✓	47.91	48.96	47.27 ± 1.30	28988	142
	PointPillars	✗	40.95	43.79	27.00 ± 1.38	34195	49
		✓	39.62	43.02	25.01 ± 1.31	28903	48
	SECOND	✗	47.38	48.29	36.88 ± 1.64	34195	52
		✓	46.08	47.59	34.73 ± 1.72	28903	50

example, in the case of PointPillars with multi-sweep input, the total number of points is reduced by more than 44 k points, which results in an approximately 39 % faster inference time. Such results are achieved while only sacrificing 1.64 mAP and 0.94 NDS percentage points. Similar results can be seen for SECOND and CenterPoint. In contrast to the ground truth prior knowledge experiments, using 2D object detections from the surrounding cameras is applicable in real-world applications. Although performing inference on multiple cameras adds to the overall computational time of the system, in multi-sensor setups like those found in autonomous driving vehicles, such detections need to be computed in any case and, therefore, do not add additional inference time overhead. Additionally, sector dropping does not require complex operations and can be efficiently computed with minimal overhead while significantly reducing the input load and inference time.

Input Reduction via Beam Sector Downsampling. In Table 4, we report the results using beam sector down-sampling instead of sector removal. The results show a similar reduction in inference time compared to the entire sector. However, both mAP and NDS yield slightly lower results. This phenomenon can be linked to possible false positive detection caused by the down-sampling artifacts. Although the down-sampling strategy is less effective than the entire sector dropping, for more advanced LiDAR sensors with more vertical layers, this could be beneficial to reduce the input load while still allowing for possible detections of objects that the 2D object detector has missed.

Table 4: Evaluation results of the proposed method using ground truth prior knowledge (Section 3.2) and beam sector down-sampling (Section 3.4) described by the parameter d. All experiments are reported using $N_S = 6$ and CenterPoint as detector. The detection metrics mAP and NDS are in percentage.

Multi Sweep	d	**mAP**	**NDS**	**Inference Time (ms)**	**Num Points**	**Num Pred Boxes**
✓	2	59.40	66.56	54.34 + 2.69	203566	131
	4	59.36	66.55	54.40 + 2.77	202117	131
	8	59.34	66.54	53.99 + 2.89	201340	131
	16	59.39	66.56	54.16 + 2.99	200947	129
✗	2	49.36	49.80	47.64 + 0.91	30165	153
	4	49.35	49.79	47.38 + 1.01	28922	150
	8	49.39	49.82	47.19 + 1.12	28306	147
	16	49.56	49.90	47.01 + 1.29	27959	144

5 Discussion

The proposed framework shows that the simple strategy of selecting point cloud regions where relevant objects are present can be highly effective in reducing the number of input points to be processed and the inference time. However, in cases where perfect ground truth prior knowledge is not available (i.e., in most real-world applications), some additional uncertainty is inevitably introduced into the perception pipeline. Although modern object detectors such as YOLOv8 achieve high performance in most situations, the failure to detect relevant objects could lead to an overall degradation of the perception pipeline. Therefore, the benefits of the proposed method need to be carefully weighed against the possible risks it can introduce when applying it in real-world applications. Nevertheless, if reliable object detection can be ensured, e.g., by the camera sensors, our framework offers the possibility to significantly reduce the computation time for LiDAR object detection with even slightly improved performance compared to the full point cloud processing with the same detectors.

6 Conclusion

In this work, we presented region dropping in LiDAR point clouds to improve the inference time of current LiDAR 3D object detectors. We exploit the fact that current autonomous vehicles are equipped with multiple sensors, such as cameras. By leveraging the 2D detections of the YOLOv8 object detector on camera images, we can find the relevant sectors in the point cloud while removing those without objects of interest. As a result, we can drastically lower the number of points that the LiDAR object detector needs to process. We evaluate our method on the large-scale nuScenes dataset. Our method is agnostic of the underlying object detector, and we evaluate it on popular LiDAR object detectors, such as PointPillars, SECOND, and CenterPoint. The results show that using the prior information obtained from the camera detections allows for a decrease in the required inference time while maintaining similar object detection performance. To showcase the

upper bound of performance gain, we simulate the benefit of perfect detections by using the ground-truth boxes for sector dropping. We hope that our research inspires new research in performance optimization in LiDAR processing, making the use of LiDAR sensors in autonomous driving more accessible.

References

[1] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLO," 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[2] Spconv Contributors, "Spconv: Spatially sparse convolution library," 2022. [Online]. Available: https://github.com/traveller59/spconv

[3] H. Tang, Z. Liu, X. Li, Y. Lin, and S. Han, "TorchSparse: Efficient Point Cloud Inference Engine," in *Conference on Machine Learning and Systems (MLSys)*, 2022.

[4] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035.

[5] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: composable transformations of Python+NumPy programs," 2018. [Online]. Available: http://github.com/google/jax

[6] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[7] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[8] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[9] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer, 2016, pp. 21–37.

[10] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[11] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp,

G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[12] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[13] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. Computer Vision Foundation / IEEE Computer Society, 2018, pp. 4490–4499.

[14] Y. Yan, Y. Mao, and B. Li, "Second: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, p. 3337, 2018.

[15] H. Tang, S. Yang, Z. Liu, K. Hong, Z. Yu, X. Li, G. Dai, Y. Wang, and S. Han, "Torchsparse++: Efficient training and inference framework for sparse convolution on gpus," in *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, 2023, pp. 225–239.

[16] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 12 697–12 705.

[17] S. Mohapatra, S. Yogamani, H. Gotzig, S. Milz, and P. Mader, "Bevdetnet: bird's eye view lidar point cloud based real-time 3d object detection for autonomous driving," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2021, pp. 2809–2815.

[18] L. Fan, X. Xiong, F. Wang, N. Wang, and Z. Zhang, "Rangedet: In defense of range view for lidar-based 3d object detection," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 2918–2927.

[19] T. Yin, X. Zhou, and P. Krahenbuhl, "Center-based 3d object detection and tracking," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 11 784–11 793.

[20] M. Henning, M. Buchholz, and K. Dietmayer, "Advancing frame-dropping in multi-object tracking-by-detection systems through event-based detection triggering," in *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*, 2023, pp. 777–782.

[21] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3d object detection from rgb-d data," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 918–927.

[22] C. R. Qi, H. Su, K. Mo, and L. 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 (CVPR)*, July 2017.

[23] Z. Wang and K. Jia, "Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 1742–1749.

[24] A. Paigwar, D. Sierra-Gonzalez, O. Erkent, and C. Laugier, "Frustum-pointpillars: A multi-stage approach for 3d object detection using rgb camera and lidar," in *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 2021, pp. 2926–2933.

[25] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11 618–11 628.

[26] O. D. Team, "Openpcdet: An open-source toolbox for 3d object detection from point clouds," https://github.com/open-mmlab/OpenPCDet, 2020.

[27] U. U. MRM, "YOLOv8 BDD100K," 2023. [Online]. Available: https://github.com/uulm-mrm/ultralytics

[28] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "Bdd100k: A diverse driving dataset for heterogeneous multitask learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2636–2645.