Can we remove the ground? Obstacle-aware point cloud compression for remote object detection

Pengxi Zeng[1], Alberto Presta[2*], Atishna Samantaray[1], Jonah Reinis[3], Dinesh Bharadia[1], Hang Qiu[4], Pamela Cosman[1]

[1] Dept. of Electrical and Computer Engineering, UC San Diego, CA, USA
[2]Computer Science Department, University of Turin, Italy
[3] Case Western Reserve University, OH, USA
[4] ECE / CSE Dept., University of California Riverside, CA, USA
[*] corresponding author, mail: alberto.presta@unito.it

***Abstract*—Efficient point cloud (PC) compression is crucial for streaming applications, such as augmented reality and cooperative perception. Tailoring compression towards perception tasks at the receiver side, we ask the question "Can we remove the ground points during transmission without sacrificing the detection performance?" Our study reveals a strong dependency on the ground from state-of-the-art (SOTA) 3D object detection models, especially on those points below and around the object. In this work, we propose a lightweight obstacle-aware Pillar-based Ground Removal (PGR) algorithm. PGR filters out ground points that do not provide context to object recognition, significantly improving compression ratio without sacrificing the receiver side perception performance. PGR is light-weight, highly parallelizable, and effective. Our evaluations on KITTI and Waymo Open Dataset show that SOTA detection models work equally well with PGR removing 20-30% of the points, with a speed of 86 FPS.**

***Index Terms*—point cloud compression, preprocessing, point removal, object detection**

I. Introduction

Point cloud streaming enables various applications, such as augmented reality, sensor fusion, and cooperative perception [1]–[3]. For example, in the context of connected and automated vehicles (CAVs), merging data from peer vehicles enhances situation awareness beyond individual on-board sensing capabilities [5]–[7]. However, storing and transmitting point clouds from LiDAR (Light Detection and Ranging) sensors is costly. A point cloud (PC) frame is represented as an array of points, each with (x, y, z) spatial coordinates and associated attributes, such as reflectance. A 64-line LiDAR generates around 100,000 points per frame, meaning that at 10 FPS scanning rate, the raw data rate of 128 Mbps exceeds the available bandwidth of current vehicle-to-infrastructure (V2I) communication technologies, such as C-V2X [8]. Therefore, it is necessary to develop efficient systems for vehicular LiDAR data transmission.

The main challenges of designing such efficient systems end-to-end are twofold: (1) a data compression technique that can fit narrow communication bandwidth in which the compressed representation is suitable for remote machine vision tasks, and (2) a low-latency compression pipeline that supports latency-sensitive machine vision tasks. To reduce data volume, previous work [26] exploits the redundancy in the raw frame to compress the PC in a lossless fashion [27], or slightly trades off the fidelity for a lower bit rate [24]. However, manipulated data in the compression process, though sometimes indistinguishable by human eyes, can significantly degrade machine vision performance. To preserve performance, machine vision models such as object detection and semantic segmentation can be used *before* quantization, but this approach only transmits incomplete or encoded information, limiting downstream tasks. In addition, these models often add significant latency, hindering real-time streaming applications. In this paper, we explore an efficient design that achieves low-latency compression without sacrificing application accuracy. Taking object detection as an example, the performance dependency on each point varies significantly. Fig. 1 shows the objects detected

Fig. 1: Object ground truth (green) and detection results (blue) using PVT-SSD with an input of original (left) PC, PC with semantic ground removal (middle, PolarNet), and PC with PGR (right). Using PC with semantic ground removal, detection bounding boxes are mismatched with ground truth (red circle), while PGR does not affect detection performance.

by PVT-SSD [25] using the original PC (left), and using the PC after removing ground points (middle) with a semantic segmentation model PolarNet [23]. It illustrates that naively removing ground points (GPs) based only on semantics may negatively impact perception performance, as state-of-the-art (SOTA) models show strong dependency on certain ground points. Our key idea is to investigate the feasibility of wisely removing irrelevant context (*i.e.,* partial ground in this case), while maintaining the performance of downstream computer vision tasks (Fig. 1 right).

Our main contributions are: (a) A feasibility and innovative study demonstrating that *a careful selection* of GPs can be removed with minimal impact on detection accuracy, (b) Experimental results on the value of retaining partial GPs close to objects, and (c) A lightweight, highly parallelizable **P**illar-based **G**round point **R**emoval (PGR) algorithm that is able to selectively retain most GPs near objects without the complexity of detecting objects.

II. Related work

A. Point Cloud Compression

A PC $X = \{\boldsymbol{p}_i \in \mathbb{R}^d\}_1^N$, is a set of N points in 3D space, where each point $\boldsymbol{p}_i$ consists of spatial coordinates x_i, y_i, z_i, and additional attributes such as RGB color and reflectance. Unlike images, this data is unstructured; the points are situated within a vast 3D space characterized by local density but sparse distribution overall. To efficiently compress a PC, it has to be first converted to a data structure that can represent position data compactly, allowing exploitation of inter-point attribute correlation. One approach uses a tree structure [22], [27], suitable for representing data with unevenly distributed point density. Draco [21] uses KD-trees for PC geometry, while MPEG [18] provides an octree-based PC compression standard (G-PCC) that we used. Alternative and more recent approaches include use of deep learning [19] or diffusion models [20].

B. Ground Point Removal

We categorize LiDAR GP removal methods into non-learning-based methods, which employ handcrafted techniques for ground segmentation, and learning-based methods, which use deep learning. In the former group, older methods such as [17] partitioned PCs to estimate GPs by comparisons with local line fits. A two-step algorithm [16] identified the ground surface iteratively by using deterministically assigned seed points and clustering the remaining non-ground points, leveraging the structure of the LiDAR PC, and [15] detected most non-ground points based on inter-ring distances, then used multi-region RANSAC plane fitting to separate the remaining non-ground. The method in [14] encoded a PC into a Concentric Zone Model–based representation, followed by ground plane fitting and ground likelihood estimation to extract the final ground segmentation.

Among learning-based methods, [10] estimated the ground plane elevation end-to-end with a grid-based representation, exploiting PointNet [9] to extract features and regressing ground height for each cell of the grid. In [23], a polar bird's-eye-view representation balanced the points across grid cells in a polar coordinate system and aligned the attention of a segmentation network with the long-tailed distribution of points along the radial axis. For segmentation, they exploited a simplified KNN-free PointNet to transform points to a fixed-length representation, and then a ring CNN that outputs a quantized prediction, decoded finally to the point domain.

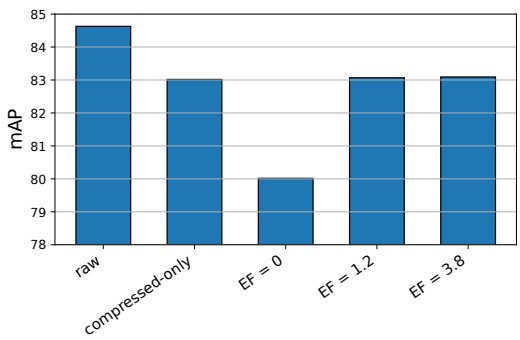


Fig. 2: Feasibility study on GP removal for Car detection in KITTI. The evaluation ranges from uncompressed 'raw' PCs to 'compressed-only' without GP removal, and then through a series of Extension Factors (EFs) plus compression.

III. Methods

A. Problem Statement

Given an input PC frame $X = \{\boldsymbol{p}_i \in \mathbb{R}^d\}_1^N$, a GP removal pre-processing algorithm f, a PC encoder g_{enc}, and a decoder g_{dec}, the encoded bitstream is $b = g_{\text{enc}}(f(X))$, and the decoded PC is $Y = g_{\text{dec}}(b)$. For a machine vision task T, with performance $p_T(Y)$, this work aims to find a suitable f that reduces bit rate significantly with negligible impact on $p_T(Y)$. We focus on the choice of f, keeping the rest fixed.

B. Impact of Ground Points on 3D Object Detection

As an initial exploration, we see how removing GPs, partially or completely, affects downstream machine vision tasks. Using input PC X, a pre-processing algorithm $f(X) = \hat{X}$ accurately removes selected GPs. PolarNet [23], a SOTA method, identifies GPs as reference for these experiments. In this first set of experiments (Fig. 2), we removed all GPs and later restored those in the extended bounding boxes of objects using ground-truth labels. The system is described as:

$$\hat{X} = f(X; S, EF) \tag{1}$$

$$Y = g_{dec}(g_{enc}(\hat{X})) \tag{2}$$

where S refers to per-point semantic segmentation from PolarNet, and EF is a factor controlling the extension from ground truth bounding boxes. For instance, for a car's bounding box sized $(length, width, height)$, EF restores GPs detected by PolarNet in a box of size $(1+EF)\times(length, width, height)$. When $EF = 0$, no extension is applied, restoring only annotated object points. The PC after pre-processing will be compressed and decompressed with the standard geometry-based

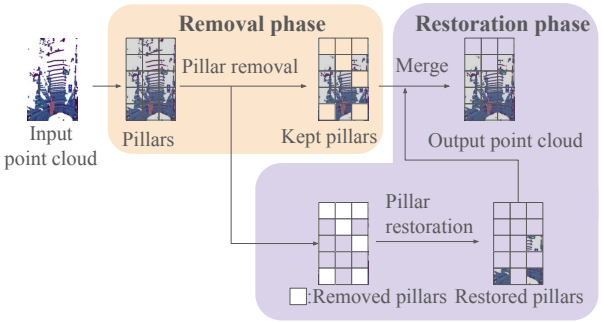


Fig. 3: Illustration of PGR with reduced number of pillars.

PC compression method G-PCC [18]. The decompressed point cloud Y is used for downstream task performance evaluation.

We exploit PVT-SSD [25] to study the performance degradation introduced by GP removal, comparing against a baseline with no removal ($\hat{X} = X$). For the Car class, full removal degrades detection accuracy by over 3% (see Fig. 2), but partial removal via $EF \in [0.3, 3.6]$ matches the detection accuracy of the baseline at lower bit rates.

C. Pillar-based Ground Removal Algorithm

Given that exploratory result that full GP removal hurts detection performance, but partial removal of GPs can save on data rates without impacting detection, we aim to construct a very low complexity algorithm that can partially remove the ground. As shown in Fig. 3, our method involves two steps, **Pillar removal** and **Pillar restoration**.

1) Pillar removal: Assuming minimal height variation in small GP regions, each input frame is divided into a 2D grid of square pillars pl_j for $j = \{1, ..., M\}$, with M total pillars. The pillar size, *resolution*, sets the algorithm's granularity. The goal is to remove pillars that are likely part of the ground. For each pillar, if the height difference between its highest and lowest points is below a certain threshold, then it might contain only GPs. For a pillar pl_i, this condition is written as:

$$d_z(pl_i) = z_{max}(pl_i) - z_{min}(pl_i) \leq \delta_{minmax} \tag{3}$$

where $z_{max}(pl_i)$ and $z_{min}(pl_i)$ are the maximum and minimum heights in the pillar, and δ_{minmax} is a threshold.

However, this condition is insufficient– if for example, a pillar passes through the roof of a car, it is possible that the height difference is small because this area is quite flat; erroneously detecting these points as ground would mistakenly remove a portion of the car. In addition to the condition on height difference, one needs to consider the neighboring area and a *local ground baseline*. We compare $z_{min}(pl_i)$ with the height b of the lowest point in a square neighborhood controlled by a parameter *environmental radius* (*er*), representing half of the side length. A pillar is considered to be ground if it fulfills condition (3) and also the following one:

$$d_{env}(pl_i) = z_{min}(pl_i) - b < \delta_{env}. \tag{4}$$

Summing up, the pillar removal step consists of an indicator function Φ_i:

$$\Phi_i = \begin{cases} 0 & \text{if } d_z(pl_i) \leq \delta_{minmax} \text{ and } d_{env}(pl_i) < \delta_{env} \\ 1 & \text{otherwise} \end{cases} \tag{5}$$

Parameters should be based on physical aspects of PCs, not specific datasets. For δ_{minmax}, it must be large enough to classify sidewalk curbs (10-20cm) and street medians as ground points without removing small pedestrians (80-95cm). We set δ_{minmax} to 40cm. *er* is set to 1.8m, matching a car's width, to avoid misclassifying a flat car roof as GPs. δ_{env} is 40cm to distinguish a flat car roof or stroller top from the ground. These values enhance algorithm robustness. A smaller *resolution* considers smaller pillars, balancing grid granularity and model speed; set to 40cm.

2) Pillar restoration: The *Pillar removal* stage removes points near objects, but as noted in Sec. III-B, GPs nearby are crucial for precise detection. At the pillar level, the *Pillar restoration* step reinstates some removed pillars. If a removed pillar pl_i is close to a retained one, preserving pl_i is beneficial for machine vision. A pillar pl_i is restored if any retained pillar pl_j is within a Chessboard distance below the threshold δ_{res}. We use R_i to denote a restoration flag indicating whether to restore pl_i:

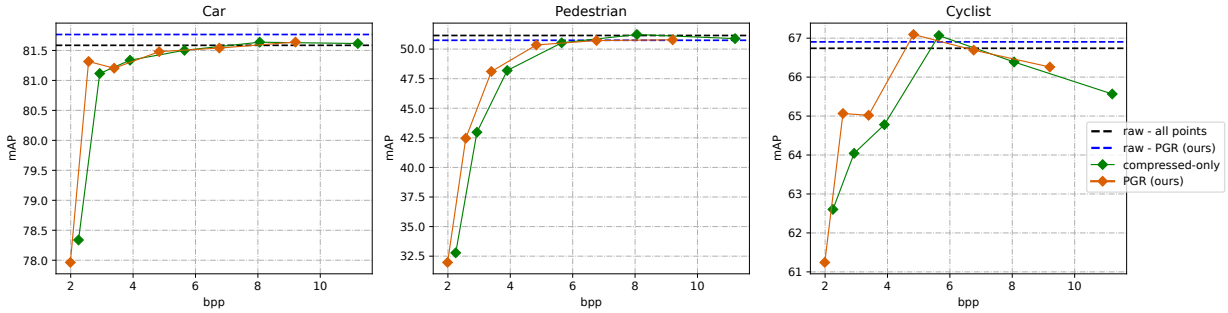
$$R_i = \begin{cases} 1 & \text{if } \sum_{j|D(i,j)\leq\delta_{res}} \Phi_j > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

As points become sparser farther from the LiDAR, objects may lose critical points that contain information about their global geometry. Such important points could be close to the ground and thus removed in the removal phase, but cannot be restored if δ_{res} is not large enough. To strike a good trade-off between removal efficiency and detection precision, we adapt δ_{res} for different ranges. From the experiments, for pillars closer than 30m, we used $\delta_{res,1} = 1.8$m for KITTI and $\delta_{res,1} = 2.2$m for the Waymo Open Dataset. For both datasets, we used $\delta_{res,2} = 5.4$m for pillars further away. In far ranges, the sparsity of points means a larger δ_{res} slightly increases the number of remaining ground points. Fig. 3 depicts the preprocessing framework, which is is not recursive and is applied only once, obtaining real-time runtime.

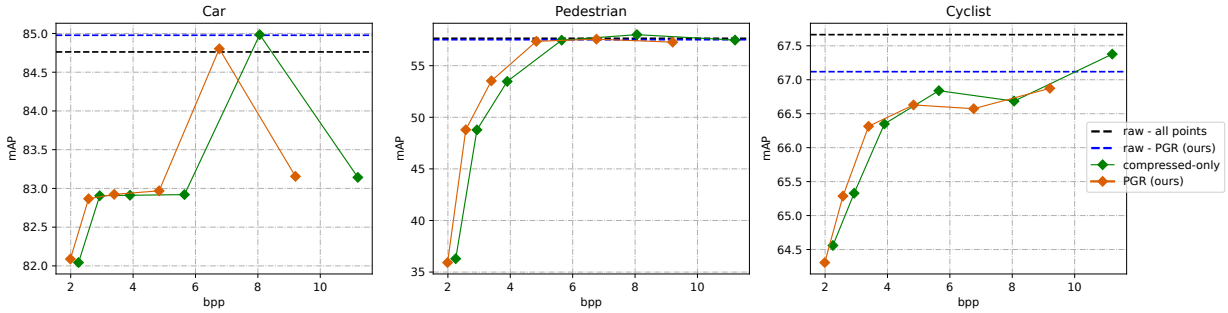
IV. Experiments

A. Experimental Setup

We chose six geometry scaling factors (0.01, 0.012, 0.015, 0.022, 0.035, 0.063) paired with six attribute quantization parameters (34, 30, 26, 22, 18, 14). The compressed PC bit rate is expressed as bits per point (bpp), calculated by dividing the total number of bits in the compressed frame by the total number of points in the PC for that frame. We use object detection to evaluate the effectiveness of PGR. The PC after applying PGR is compressed and decompressed and subsequently sent through pre-trained object detection models. We used **Mean Average Precision (mAP)** as the detection performance [13].



(a) mAP vs bpp using SECOND.



(b) mAP vs bpp using PVT-SSD.

Fig. 4: mAP vs bpp on KITTI dataset using (a) SECOND and (b) PVT-SSD. Dotted lines represent results on uncompressed data. For mAP, higher values indicate better performance, whereas for bpp, lower values are preferable.

B. Datasets

1) KITTI: The KITTI dataset [12] includes 7481 training samples and 7518 test samples, usually split into a training set of 3712 samples and a validation set of 3769 samples. Objects in the camera's field of view are annotated, categorized as Car, Pedestrian, and Cyclist, and classified into easy, moderate, and hard levels based on bounding box size, occlusion, and truncation.

2) Waymo: The Waymo Open Dataset (WOD) [11] offers 798 training and 202 validation scenes, with separate LiDAR and camera annotations. Objects are classified into difficulty LEVEL_1 for over five points, and LEVEL_2 for at least one point. LiDAR entries include (x, y, z) coordinates, intensity, and elongation. Frames may have a *no-label zone* without annotations due to distance or irrelevance, so we exclude these regions to avoid impacting detection performance. WOD frames, originally in range image format with distance, intensity, and elongation as pixel values, were converted to a 3D format with 32-bit float coordinates and attributes.

C. PGR Performance on Object Detection

1) bpp vs. mAP: Figures 4 and 5 display bpp vs. mAP for two datasets. Solid lines indicate results for different compression levels, with orange for PGR and green for the full PC (no GPs removed). Dashed lines show results from the uncompressed dataset. Although the rate reduction depends on the operating point, PGR effectively shifts the rate-performance curve to the left, generally reducing the bitrate; this means we can achieve the same desired task performance with fewer bits. In Fig. 4, the PGR curve is above the non-processed curve for most of the range, for both models and all objects classes. For Waymo (Fig. 5), there are gains for the Pedestrian and Cyclist classes, while for the Vehicle class, our curve is slightly below the compressed-only baseline. At lower bpp, improvements suggest applying PGR can be beneficial in the low bit rate regime. The fraction of points that PGR removes varies by frame, from 10% in dense urban areas to 50% in open ground.

Table I shows the percentage of points that PGR preserves belonging to each category of detected object as well as overall; PGR effectively removes many GPs with minimal removal of points belonging to relevant objects. In Fig. 5, at the lowest bit rate, PGR saves 9.34% (0.282 bpp) with minor mAP changes: -0.209% (Vehicle), -0.003% (Pedestrian), -0.154% (Cyclist). At the highest bit rate, it is reduced by 12.94% (1.685 bpp) with mAP changes of -0.175%, $+0.011\%$, and -0.078% for the respective classes.

2) Run-time: On a GeForce RTX 3090 with 8 CPU cores, the speed to process frames sequentially (one by one), can reach 86.5 frames per second (fps), 11.6 ms per frame. This leaves ample latency budget for downstream tasks such as object detection, prediction, and segmentation.

D. Robustness with respect to parameter tuning

We evaluate the performance when varying the parameters introduced in Sect. III. The parameters have specific real-world physical meanings that remain consistent across datasets (KITTI and Waymo) and object detection models. Fig. 6

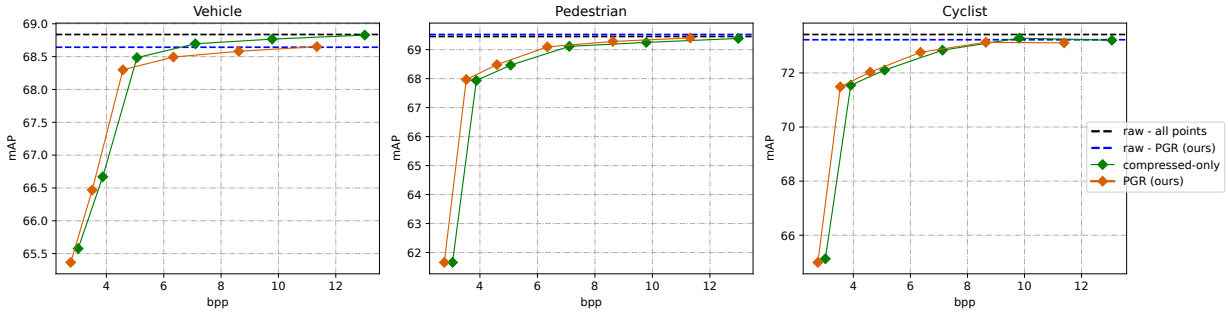


Fig. 5: mAP vs bpp on Waymo Open Dataset using PVT-SSD. Dotted lines represent results on uncompressed data.

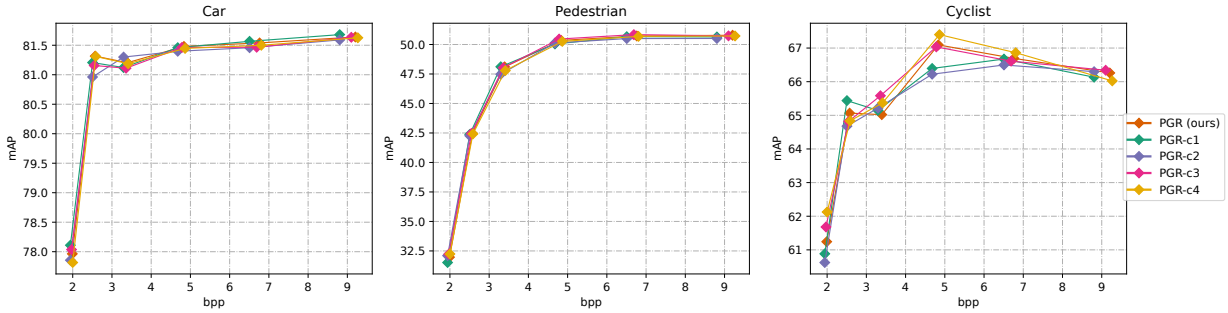


Fig. 6: mAP vs bpp on KITTI Dataset using SECOND, considering PGR with different configurations

TABLE I: Avg Percentage (%) of points preserved by PGR within each category of detected object as well as overall

Points Category	**KITTI**	**Waymo**
Vehicle	99.981	99.980
Pedestrian	99.995	99.982
Cyclist	99.995	99.988
Overall	75.133	82.415

TABLE II: BD-mAP of our PGR against SOTA for GP removal used as pre-processing and PVT-SSD as pre-trained model. We consider *compressed-only* as anchor.

	Car	Pedestrian	**Cyclist**
	BD-mAP	BD-mAP	BD-mAP
PGR (ours)	**-0.14**	**1.75**	**0.008**
PGR (ours) w/o restoration	-1.89	**4.36**	-0.57
GnDNet	-4.16	-13.72	-6.388
PolarNet	-1.42	-0.427	-0.95
patchwork++	-1.38	-0.97	-0.949

shows the results of mAP vs. bpp on the KITTI dataset using SECOND, considering different configurations of our method. The orange line (PGR-c0) represents the standard configuration introduced in Sect. III, while in the other scenarios we manually changed parameters as follows:

- PGR-c1: *er* passes from 1.8 to 1.4.
- PGR-c2: $\delta_{res,1}$ passes from 1.8 to 1.4.
- PGR-c3: δ_{minmax} = 0.6.
- PGR-c4: (*er*, δ_{minmax},$\delta_{res,1}$,$\delta_{res,2}$) = (0.6,0.35,1.6,5.2)

Here we mention only the values that have been changed. The mAP remains similar in all configurations; for both *car* and *pedestrian* the differences are negligible, with only modest variations for the cyclist in the central bit range, without catastrophic degradation. Even for PGR-c4, where we modified 4 parameters, the final performance does not change remarkably, demonstrating the robustness of PGR to parameter tuning. The configurations for KITTI and Waymo in Sec. III are essentially the same, showing consistent performance in all datasets without significant parameter adjustments.

E. Comparison with other ground point removal methods

We compare PGR with other ground removal methods used for pre-processing, namely PolarNet [23], GndNet [10], (deep-based methods) and patchwork++ [14] (handcrafted method). Table II shows results for KITTI using PVT-SSD as the pretrained model for object detection. We used the Bjontegaard [4] metric with mAP. Although BD-mAP is rare in object detection, it effectively summarizes results across various compression levels. PGR outperforms all models in every object class, equaling or surpassing the compressed-only anchor's performance, where ground points remain. Fig. 7 illustrates these results for the Cyclist class. These results align with Sec. III-B, showing that removing all GPs degrades performance. While other methods remove all GPs, PGR maintains object detection performance by selectively removing GPs. Despite PGR preserving 75% of ground points in KITTI and performing poorly in ground detection, it achieved the best object detection results.

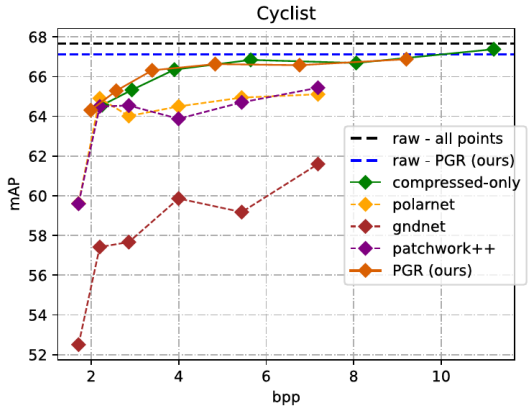


Fig. 7: mAP vs bpp of PGR against other ground point removal methods considering Cyclist class.

F. Ablation Study

Tab. II shows results with and without the PGR restoration phase. This phase aims to restore some points that were mistakenly removed in the algorithm's first step. Despite an obvious slight gain in terms of bits saved, we observe a significant decrease in precision, especially for the Car and Cyclist classes. Without the restoration phase, we retain 98.076% of car points, 98.584% of cyclist points, and 98.844% of pedestrian points, values closely matching those in Table I with restoration; the latter class is the only one that presents some degradation after the restoration phase. This suggests that for smaller objects, detection models may require fewer ground points around them to provide context, and that with a more compact geometry, critical points are less likely to be lost during the removal phase.

V. Conclusion and Future Works

PGR highlights the necessity of retaining certain ground points in PCs for remote detection of cars, pedestrians, and cyclists. We introduce a method to eliminate unnecessary ground points. Drawing on Section III-B, which suggests ground near objects aids machine vision, we devised a two-step algorithm: remove likely ground points, then restore those near objects. Reference PGR achieved excellent bit rate reduction without losing detection precision, especially at low bit rates, and PGR achieved real-time performance at 86.5 fps. In Sec. IV-E, our method proved more effective than other SOTA techniques offering superior results for all tested objects, while in Sec. IV-F we analyzed the impact of the restoration phase. Future work will aim to improve the algorithm's resilience to uneven surfaces and small objects, and refine parameter selection through Bayesian optimization.

References

[1] Qiu, Hang et al. "AutoCast: scalable infrastructure-less cooperative perception for distributed collaborative driving." In *Proceedings of the ACM MobiSys*, 2022

[2] Qiu, Hang et al. "AVR: Augmented vehicular reality." In *International Conference on Mobile Systems, Applications, and Services*, 2018

[3] Chen, Q. et al. "F-cooper: Feature based cooperative perception for autonomous vehicle edge computing system using 3D point clouds." In *IEEE Symposium on Edge Computing*, 2019

[4] Bjontegaard, Gisle. "Calculation of average PSNR differences between RD-curves." In *VCEG-M33*, 2001

[5] Wang, Tsun-Hsuan et al. "V2VNet: Vehicle-to-vehicle communication for joint perception and prediction." In *European Conference on Computer Vision*, 2020

[6] Xu, Runsheng et al. "CoBEVT: Cooperative Bird's Eye View Semantic Segmentation with Sparse Transformers." In *Conference on Robot Learning*, 2023

[7] Xu, Runsheng et al. "V2X-ViT: Vehicle-to-everything cooperative perception with vision transformer." In *European Conference on Computer Vision*, 2022

[8] Cui, Jiaxun et al. "Coopernaut: End-to-end driving with cooperative perception for networked vehicles." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022

[9] Qi, Charles R. et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017

[10] Paigwar, Anshul et al. "GndNet: Fast ground plane estimation and point cloud segmentation for autonomous vehicles." In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020

[11] Sun, Pei et al. "Scalability in perception for autonomous driving: Waymo open dataset." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020

[12] Geiger, Andreas et al. "Are we ready for autonomous driving? the KITTI vision benchmark suite." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2012

[13] Everingham, Mark et al. "The Pascal Visual Object Classes Challenge: A Retrospective." In *International Journal of Computer Vision 111*, 2015

[14] Lim, Hyungtae et al. "Patchwork: Concentric zone-based region-wise ground segmentation with ground likelihood estimation using a 3D LiDAR sensor." In *IEEE Robotics and Automation Letters*, 2021

[15] Narksri, Patiphon et al. "A slope-robust cascaded ground segmentation in 3D point cloud for autonomous vehicles." In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2018

[16] Zermas, Dimitris et al. "Fast segmentation of 3d point clouds: A paradigm on lidar data for autonomous vehicle applications." In *IEEE International Conference on Robotics and Automation (ICRA)*, 2017

[17] Himmelsbach, Michael et al. "Fast segmentation of 3D point clouds for ground vehicles." In *IEEE Intelligent Vehicles Symposium*, 2010

[18] ISO/IEC JTC 1/SC 29/WG 7. "G-PCC codec description v12." In *ISO/IEC*, 2021

[19] Wang, Jianqiang, et al. "Lossy point cloud geometry compression via end-to-end learning." In *IEEE Transactions on Circuits and Systems for Video Technology*, 2021

[20] Spadaro, Gabriele et al. "Denoising Diffusion Probabilistic Model for Point Cloud Compression at Low Bit-Rates." In *IEEE International Conference on Multimedia & Expo*, 2025

[21] Huang, Tianxin et al. "3D point cloud geometry compression on deep learning." In *Proceedings of the 27th ACM International Conference on Multimedia*, 2019

[22] Elseberg, Jan et al. "One billion points in the cloud–an octree for efficient processing of 3D laser scans." In *ISPRS Journal of Photogrammetry and Remote Sensing 76*, 2013

[23] Zhang, Yang et al. "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*, 2020

[24] Schwarz, Sebastian, et al. "Emerging MPEG Standards for Point Cloud Compression." In *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2019

[25] Yang, Honghui et al. "Pvt-ssd: Single-stage 3d object detector with point-voxel transformer." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023

[26] Graziosi, Danillo et al. "An overview of ongoing point cloud compression standardization activities: Video-based (V-PCC) and geometry-based (G-PCC)." In *APSIPA Transactions on Signal and Information Processing*, 2020

[27] Schnabel, Ruwen et al. "Octree-based Point-Cloud Compression." In *PBG@SIGGRAPH*, 2006