

Article

CACTUS: Content-Aware Compression and Transmission Using Semantics for Automotive LiDAR Data

Daniele Mari , Elena Camuffo  and Simone Milani * 

Department of Information Engineering, University of Padova, Via Gradenigo 6/A, 35131 Padova, Italy; daniele.mari@dei.unipd.it (D.M.); elena.camuffo@dei.unipd.it (E.C.)

* Correspondence: simone.milani@dei.unipd.it

Abstract: Many recent cloud or edge computing strategies for automotive applications require transmitting huge amounts of Light Detection and Ranging (LiDAR) data from terminals to centralized processing units. As a matter of fact, the development of effective Point Cloud (PC) compression strategies that preserve semantic information, which is critical for scene understanding, proves to be crucial. Segmentation and compression have always been treated as two independent tasks; however, since not all the semantic classes are equally important for the end task, this information can be used to guide data transmission. In this paper, we propose Content-Aware Compression and Transmission Using Semantics (CACTUS), which is a coding framework that exploits semantic information to optimize the data transmission, partitioning the original point set into separate data streams. Experimental results show that differently from traditional strategies, the independent coding of semantically consistent point sets preserves class information. Additionally, whenever semantic information needs to be transmitted to the receiver, using the CACTUS strategy leads to gains in terms of compression efficiency, and more in general, it improves the speed and flexibility of the baseline codec used to compress the data.

Keywords: point clouds; LiDAR; compression; transmission; semantic segmentation



Citation: Mari, D.; Camuffo, E.; Milani, S. CACTUS: Content-Aware Compression and Transmission Using Semantics for Automotive LiDAR Data. *Sensors* **2023**, *23*, 5611. <https://doi.org/10.3390/s23125611>

Academic Editor: Hai Dong

Received: 28 April 2023

Revised: 8 June 2023

Accepted: 13 June 2023

Published: 15 June 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Sensing technologies have been recently boosted by the availability of versatile 3D acquisition devices that have proved to be capable of generating accurate models of the surrounding scene at a reasonable cost and in a limited time. Within these scenarios, LiDAR technology has played a leading role considering its widespread application in several fields including (but not limited to) land sensing and reconstruction, robotics, cultural heritage, and monitoring [1–3]. With respect to automotive applications, LiDAR PCs allow a real-time 3D modeling and semantic understanding of the environment [4] which enable smart applications such as autonomous driving, traffic monitoring, and path planning, to mention some of them.

However, most data processing architectures are unable to elaborate efficiently a large volume of LiDAR data in a reasonable time [5]. In addition, sharing environmental information could be required in order to enable a more efficient scheduling of operation plans, reorganize navigation paths, or access more efficient classification tasks [6,7]. It is possible to overcome this bottleneck by employing some shared highly performing computational resources, moving most of the advanced processing toward cloud computing environments [8–11].

Moreover, transferring part or most of 3D processing operations from terminals to network processing units implies transmitting big amounts of information, which must preserve its informational and semantic contents in order to be useful for the considered elaboration task (e.g., object classification, semantic segmentation, planning, etc.). Compression proves to be crucial since it allows constraining the impact on computing and

transmission facilities (e.g., latency, network congestion, and energy consumption). Indeed, whenever coding operations are too aggressive or unaware of final application, the semantic information of the scene cannot be recovered any more as the data arrive at the processing unit (see results on semantic segmentation reported in Section 4). The compression of terrestrial LiDAR point clouds is a notoriously hard problem in 3D data transmission due to the spatial sparsity and inhomogeneity of acquired 3D points as well as to the noise that affects their coordinates.

For these reasons, aside from the most known PC codecs such as TMC13 [12] and Draco [13], several other approaches have been proposed. As an example, the work by Huang et al. [14] uses an entropy model to exploit the remaining redundancy after an octree decomposition of voxel volume [15]; Tu et al. [16] use Recurrent Neural Networks to progressively encode residuals, while in the work by Varischio et al. [17], the most meaningful parts of the PC are transmitted after a segmentation of the input PC. Other meaningful approaches that are not specifically targeted toward LiDAR but that are worth mentioning are deep learning (DL)-based Point Cloud Coding (PCC) approaches [18–24] that generally exploit convolutional autoencoders for efficient features extraction and decoding.

Following these trends, some preliminary works on multimedia compression and transmission have highlighted the fact that cognitive and semantic information can be extremely useful when applied to source coding and multimedia data transmission [25].

The advantage of a semantic-aware coding system is two-fold:

1. Semantic information allows dynamically allocating the bit rate to different parts of the PC and schedules information transmission in a content-aware manner (while discarding useless data).
2. The distortion introduced by coding operations could prevent a correct classification when data are reconstructed and segmented at the receiver (see Section 4); a cognitive source coding approach allows signaling to the receiver semantic information obtained from the originally acquired data.

Most of the works that were explored in the literature address the compression task in a very general way and thus do not exploit semantic data for transmission, since it is not always available. To the best of our knowledge, up to the publication of this paper, the only work that makes use of this additional information is [17]. However, the authors fail to fully exploit it since, in the paper, semantics only serve the purpose of selecting pieces of the PC that should be discarded in order to obtain lower compression rates and faster decoding. As a matter of fact, they do not realize that the pre-computed semantic information can be transmitted together with the geometry almost for free. Additionally, in their case, the semantic information is used to divide the PC very coarsely, but increasing granularity does not really hurt performance and actually provides greater flexibility. For these reasons, we believe that the advantages of using semantic information for guiding compression and transmission still have not been properly explored.

In a nutshell, the current paper proposes a general framework for LiDAR PC compression that first segments the input PC into multiple semantically characterized clusters that are then independently coded and transmitted, tuning the compression parameters according to the significance of the coded class and the desired quality level.

This approach is particularly well suited for LiDAR data, since it is one of the most used sensors in automotive scenarios where the Semantic Segmentation (SS) literature is already well established.

The main novelties and advantages of the proposed approach can be listed as follows:

- Semantic information can be transmitted with a very limited overhead (see Section 3), sparing the segmentation task at the decoder side.
- Transmitting labels proves to be more accurate than operating SS at the decoder on the reconstructed PC (because of compression noise).
- The bitrate can be flexibly allocated by choosing the quality level or skipping some parts in a content-aware manner according to the network conditions or user preferences.

- The execution time required to compress the PC and the semantic information is lower with CACTUS than with the reference codecs. This result is obtained without considering the time required to compute the semantic information, because this step needs to be computed regardless of the coding procedure.

The framework was implemented using RandLA-Net [26] for the segmentation part, and TMC13 [12], Draco [13], and Distributed Source AutoEncoder (DSAE) [24] for the compression part. Nonetheless, the processing pipeline can be generalized to different SS architectures and codecs. The architectures choice was mainly driven by the fact that the considered algorithm is intended for a mobile automotive scenario, where latency and computational complexity are significantly constrained. In this regard, experimental performances were tested on a subset of SemanticKITTI [27,28], which is one of the most popular benchmarks for autonomous driving.

The paper is organized as follows: in Section 2, the main advancements in collaborative strategies using LiDAR PCs are presented together with an explanation about PCs codecs; then, in Section 3, the proposed method is introduced and analyzed while its performances are examined in Section 4 and all the results are further discussed in Section 5. Finally, in Section 6, the conclusions are drawn, which are followed by some hints for future works. The code for the framework can be found at <https://github.com/Dan8991/CACTUS-Content-Aware-Compression-and-Transmission-Using-Semantics-for-Automotive-LiDAR-Data> (accessed on 15 January 2023).

2. Related Work

Due to the increasing hype that self-driving cars are rising in the industry and academia, a lot has been accomplished to reduce at its minimum the computational and transmission cost of the acquired data. In particular, collaborative strategies have been developed to avoid redundant calculations, while coding algorithms have been designed in order to produce representations that are as efficient and effective as possible, facilitating data exchange among nearby vehicles.

2.1. Collaborative Strategies for LiDAR Point Clouds

The idea of sharing LiDAR PC data and processing them in a collaborative and distributed fashion has been recently investigated in many applications ranging from cultural and natural preservation [2,29,30] to automotive applications [31,32].

In [11], Shin et al. propose a road object mapping system for LiDAR data based on an edge-fog-cloud platform. Similarly, it is possible to obtain an HD mapping of the surrounding environment by fusing multiple contributions acquired by diverse vehicles at different locations and instants [33]. The problem of registering and fusing multiple acquisitions (which can have different orientations and distances) can be solved by matching lines and geometrical properties [1] or analyzing semantic information [34,35]. Alongside with reconstruction problems, the approach [10] tackles the problem of localizing the mobile terminal in an indoor environment thanks to an edge-side Simultaneous Localization And Mapping (SLAM) engine. A similar problem is solved in [36] for a swarm of unmanned ground vehicles (UGVs) using a two-stage graph optimization. Similarly, merged LiDAR data are employed for multiple object detection and tracking combining estimation filtering with a discretization of space in [37]. In addition, the solution in [38] adopts some a priori information about the environment and the context to localize cars using some low-complexity estimation strategies.

Together with reconstruction and localization, the semantic analysis of the surrounding environment benefits from collaboration as well. During the last few years, it has become possible to find different collaborative perception and classification frameworks [39–41] showing the robustness and the safety level that can be obtained by such applications. Alongside a distributed orchestration of the operations, these collaborative approaches integrate different Deep Learning data processing techniques. The approach in [42] resorts to a reinforcement learning algorithm to create a map of the environment, while a position-

channel attention mechanism is adopted in [43] to enhance some LiDAR-related features in order to discard the least-relevant information. A transformer-based architecture is employed in [44] on multispectral LiDAR data for land classification.

Although most of the approaches imply the transmission of PCs to the central processing framework, some solutions resort to federated learning strategies to avoid disseminating the data sensed by the different terminals with all the privacy-related implications [45].

2.2. An Overview of Compression Strategies for LiDAR Point Clouds

In the literature, PCC approaches can be divided into two separate classes: traditional source coding solutions [12,46,47] and DL-based codecs [18–23,48–50]. In the first set of approaches, a compact representation of the geometry is obtained by using an ad hoc data structure (e.g., a graph, a KD-tree [51], an octree [15] or cellular automata [52]) and then entropy coding. Some recent solutions aim at exploiting the temporal redundancy existing between temporally adjacent acquisitions by entailing a bidirectional prediction [53].

In the second set of approaches, instead, deep auto-encoders are usually exploited to generate smaller and entropy-efficient hidden feature representations [18–22,24].

In the work by Huang et al. [14], the PC is quantized and encoded using an octree decomposition where each set of octants can be represented by a byte. Such representation proved to work well on LiDAR PCs since sparsity can be handled efficiently. A deep entropy model is then used to estimate the probability of each symbol, and the resulting values are used as context by an arithmetic coder.

In [16], the PC is represented in a format similar to depth maps. This 2D representation is then compressed with an autoencoder, where a quantizer block maps the latent space features into binary strings. In this scheme, the Neural Network (NN) performs a first compression of the original image; then, some residuals are progressively encoded to refine the accuracy of the reconstructed samples. An LSTM-based solution is proposed in [54] obtaining higher performances with respect to other state-of-the-art PC codecs. However, such approaches are computationally burdensome and do not allow for content-aware compression.

In the work by Varischio et al. [17], Draco [13] is used in conjunction with Rangenet++ [55] to perform the encoding. In particular, the authors define three compression levels where parts of the PC are progressively removed to allow to perform compression in real time. At the highest quality level, the whole PC is coded; at the second one, points related to the street are removed, and at the lowest quality level, only points for critical classes (e.g., person, car, motorcycle) are retained. This codec allows shaping the bitrate according to the network constraints, but the bit allocation may be non-optimal whenever data from all the classes (e.g., road and vegetation) are required. On top of that, semantic labeling is discarded during transmission so the receiver has to recompute it himself on the reconstructed point cloud. The latter has holes and is affected by compression noise; therefore, it is likely to lead to a less accurate semantic segmentation.

3. Methodology

The CACTUS pipeline is composed of two main building blocks. The first is designed for a semantic understanding and organization of the acquired samples representing the scene, while the second consists in the core compression strategy that, adequately optimized, is going to create the bit stream. The proposed framework can be represented by the block diagram reported in Figure 1, where the input PC is segmented into separate semantic classes that are independently coded by a PCC architecture. Indeed, in the CACTUS framework, we assume that a SS unit is available on automotive terminals, since it is employed for the navigation or safety issues (e.g., pedestrian detection, obstacle avoidance, et.). As a matter of fact, the acquired point clouds could be semantically segmented and characterized after their acquisition.

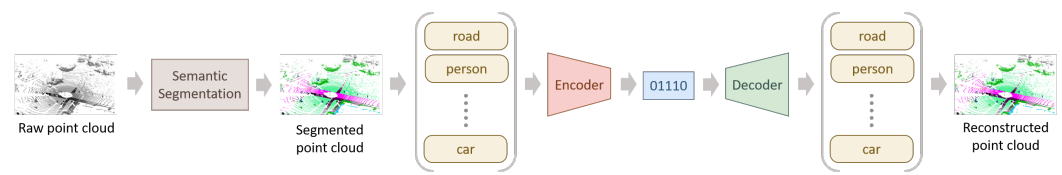


Figure 1. Schematic representation of CACTUS coding process.

In the current implementation, we adopted the architecture RandLA-Net for SS, while standardized codecs TMC13 or Draco were considered for the compression of the acquired point clouds and the formatting of the transmitted bit stream. These segmentation and coding architectures were chosen since they prove to be among the most suitable and standard-compliant solutions for LiDAR point cloud coding, although several other codecs or segmenting networks [56,57] could be adopted as the proposed framework can be generalized very well. Additionally, in order to assess the properties of the framework when using one of the most recent learned codecs, some tests were also carried out with the DSAE [24] network. The data employed belong to the validation sequence of the SemanticKITTI dataset (the first 100 samples of sequence 08).

3.1. Adopted Segmentation and Coding Solutions in the CACTUS Framework

RandLA-Net: RandLA-Net [26] is probably one of the best-performing architectures for LiDAR SS that is completely point based. In addition, it is one of the most lightweight solutions available, providing good results without needing to voxelize the input data. RandLA-Net tackles the problem that many architectures have in capturing point-wise context information as it proposes an effective local feature aggregation module, which is able to automatically preserve complex local structures by progressively increasing the receptive field for each point. This module can obtain successively larger receptive fields by explicitly considering the local spatial relationship and point features, being more effective and robust for learning complex local patterns. In general, RandLA-Net only relies on random sampling within the network, thereby requiring much less memory and computation. The entire network only consists of shared multi-layer perceptrons, making it ideal for large-scale point clouds.

TMC13: TMC13 is an implementation by the MPEG group of the Geometry-based Point Cloud Coding (GPCC) standard; it works by encoding the PC with an octree that is then processed by an arithmetic coder to reduce redundancy. The main way to perform rate control is to quantize the coordinates before entropy coding. In particular, the *codingScale* parameter can be used to control the amount of quantization. In the results section, we refer to a quantization parameter qp where the two are related by the equation $codingScale = 2^{-qp}$. In TMC13, the attributes are encoded using the Region Adaptive Hierarchical Transform (RAHT) transform [58]. In this work, the class is encoded as the color attribute by setting the values r_i, g_i, b_i relative to point $p_i = (x_i, y_i, z_i)$ as its semantic class c_i . This was achieved because, to the best of our knowledge, arbitrary attribute compression is not supported in TMC13 and Draco, but this procedure should be almost equivalent to an ad hoc implementation since the arithmetic coder should be able to completely remove the redundancy due to the repetition of the value over the three channels being deterministic.

Draco: Draco is an efficient codec implemented by Google that performs compression based on KD-trees [51] instead of octrees. This data structure has the advantage that it can place the boundaries that split the space where it is most efficient, which allows to better encode point clouds with uneven densities.

Draco with respect to TMC13 has the advantage that it provides the *cl* (compression level) parameter that can be used to reduce the computation time at the price of worse compression ratios. Here, the qp parameter was computed as $qp = 15 - qp_d$, where qp_d is the quality parameter defined by Draco. In this case, the class was losslessly encoded as the color likewise TMC13.

DSAE: DSAE is a learned geometry codec proposed in [24] based on the principles of distributed source coding [59]. The main idea of this paradigm is that the receiver knows

a signal $\hat{x}$, called side information, that is correlated with a signal x at the sender side. Because of this, the latter can just transmit some bits s called syndromes that the receiver can use to correct his own signal, increasing its similarity with x . In this case, the side information is represented by the PC compressed with another codec (TMC1 in the original paper). During transmission, the sender will use a convolutional encoder to compute the syndromes, which will be compressed with an arithmetic coder and transmitted to the receiver. The latter will feed them, together with the side information, to a convolutional decoder that should be able to reconstruct the PC with higher quality with respect to the side information.

For a more detailed description of the approach, we refer to the original paper.

3.2. Content-Aware Compression for Automotive LiDAR Point Clouds

This subsection presents the operational pipeline that defines the general backbone of the CACTUS coder. The input point cloud $P = \{p_1, \dots, p_n\}$ is processed by the semantic segmentation unit (RandLA-Net), splitting the full acquisition into k subsets $P_i = \{p_{i,1}, \dots, p_{i,n_i}\}$, $i \in \{0, \dots, k\}$, where the gathered points belong to the same semantic class $c_{i,j} = i \forall j$. Empty segments, i.e., such that $|P_i| = 0$, are skipped. Whenever $|P_i| \neq 0$, the point subset is compressed using a configurable PCC generating a binary stream b_i . Codec parameters for P_i can be manually tuned (e.g., the Quantization Parameter (QP)) in order to vary the reconstruction accuracy and the allocated bitrate. This larger amount of degrees of freedom makes the approach more flexible with respect to [17], where three fixed configurations are adopted and only a limited set of information is transmitted. In this way, the user or the algorithm can choose to lower the quality of semantic classes (or even completely remove them) that are less useful in the considered scenario (e.g., a self-driving car might not need to precisely know the position of the vegetation outside the road). This allows the system to be more robust to scenarios when the bandwidth is limited, for example when there is high network congestion.

Moreover, the CACTUS scheme allows transmitting semantic class information alongside with point cloud geometry data, making semantic segmentation unnecessary at the receiver side. The compressed file format can be represented by Figure 2 and consists of a small initial header h , which is followed by the sizes $(s_1, \dots, s_k)$ in bytes of each compressed representation and the corresponding encoded points. Therefore, it is possible to represent the final bit stream with the concatenation $b = \text{concat}([h, s_1, b_1, s_2, b_2, \dots, s_k, b_k])$. The header h is formatted in a 4-byte string, where each bit flags whether the associated semantic class is present in the bit stream or not (in the case of SemanticKITTI, only 20 bits are strictly required). The integer s_i is coded by 4 bytes and represents the size of the compressed PC b_i . The total size of the bitstream can be computed as:

$$\text{len}(b) = \text{len}(h) + \sum_{i=1}^k (\text{len}(s_i) + \text{len}(b_i)) \quad (1)$$

where the $\text{len}(\cdot)$ function returns the size in bytes of the argument. Adding the header h allows sending semantic information just with an overhead of $4(k+1)$ bytes, which is almost negligible with respect to the total transmission size. This way, the decoder can obtain all the required information to reconstruct the full PC. In fact, using the file sizes, it can obtain the compressed version of the segments that can be independently decoded by TMC13, Draco, or DSAE without any additional information. Additionally, thanks to the binary flags in the header, it can easily infer the class relative to each PC P_i .

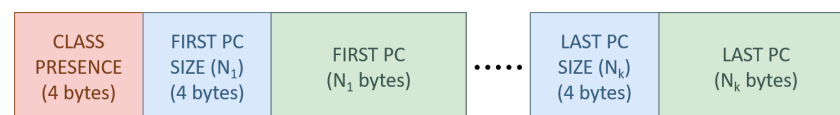


Figure 2. Structure of the PC compressed with the CACTUS.

Note that the proposed architecture is quite general, since any segmentation or compression algorithm can be used depending on computational and accuracy requirements. Experimental results showed that the coding performance is not significantly affected by the choice of the SS algorithm. Indeed, even if some points are misclassified (thus becoming isolated after partitioning the PC), many codecs have some coding options that efficiently handle this eventuality (e.g., direct coding mode in TMC13). On the contrary, compression performance can significantly depend on the codec's robustness with respect to sparsity and noise. For this reason, experimental data were obtained on TMC13 and Draco when considering standard codecs and with DSAE for learned approaches.

3.3. Learned Point Cloud Coding Methods

Before moving to the results section, it is important to briefly explain how DSAE was integrated in CACTUS and to mention the main problems that Learned Point Cloud Geometry Codecs (LPCGCs) approaches currently display when used in CACTUS or more in general when applied to LiDAR PCs.

While Draco and TMC13 are widely tested approaches that are very robust to sparsity and other issues that often affect PCs, the same cannot be said for Deep Learning-based approaches. As a matter of fact, many of these need to work on voxel grids in order to be able to process the data, since this allows for the usage of 3D convolutions. This is, however, not well suited for LiDAR PCs where in the regions close to the sensor the signal is very dense, while when further away, it is very sparse. This means that using a coarse voxel grid would destroy a lot of information close to the center, while using a finer resolution would result in very sparse regions that convolutional neural networks struggle to handle and that would increase the bitrate excessively.

This problem is tackled in TMC13, for example, by using the direct coding mode, but these techniques are not yet implemented in learned codecs, since they are still at very early stages, and researchers are still trying to solve many of the challenges presented by PCs signals.

All the aforementioned problems are slightly amplified by the CACTUS framework, since it further increases the sparsity because of the partitioning induced by SS.

Finally, the most widespread LPCGCs are trained on datasets obtained by sampling the points from a mesh (e.g., [21,22] train on ModelNet40 [60]) or on PCs obtained with stereo systems such as the ones from the MPEG dataset (e.g., [18,23,24]), and very few works focus specifically on LiDAR data. For this reason, most of the LPCGCs suffer from the shift in the input distribution when tested on this scenario.

Ideally, the best way to exploit these codecs with CACTUS would be to train a conditional model that exploits information about the class of the partitioned points to achieve better rate-distortion performance; however, this would require an ad hoc codec which goes against the philosophy of CACTUS that was designed to work without major modifications with most codecs. For this reason, in this work, we use the DSAE codec proposed in [24], with some slight modifications, as the main codec for the system. The main differences with respect to the original paper are two: i.e., TMC13 is used instead of TMC1 to generate the side information (which provides higher quality side information at a lower rate). Second, the syndromes are forced to be binary by applying sigmoid with the addition of uniform noise $\mathcal{U}(-0.5, 0.5)$ to force the network to predict either 0 or 1, since this allowed obtaining a similar performance while making it easier for the arithmetic coder to compress the bitstream.

The main advantage that DSAE has with respect to other learned codecs is that it exploits side information encoded with another codec (in this case, TMC13). For this reason, even in very sparse regions, it is still able to reconstruct some points (which other learned codecs fail at doing) and with a very small overhead, since only a few bits of syndromes need to be transmitted.

One final issue with learned point cloud codecs is that they usually do not implement attribute compression. This is due to the fact that it is usually very hard to condition

the attributes on the decoded geometry, which often results in unsatisfactory results. Additionally, in this scenario, we would like the compression of the attributes to be lossless. For these reasons, in order to compare the compression performance of CACTUS with that of DSAE, we implement attribute compression by means of RAHT (similarly to what is used in TMC13). All the other details such as the packet structure are the same as above.

4. Experimental Results

In both TMC13 and Draco, lossy coding is performed via coordinates quantization, whose intensity is ruled by parameters *codingScale* (quality–rate trade-off) and qp_{draco} (quality), respectively. In order to equalize the testing conditions between the two codecs, a generalized quantization parameter qp is used in the experimental tests. Consequently, $codingScale = 2^{-qp}$ for the TMC13 codec, while $qp = 15 - qp_{draco}$ for Draco. Since DSAE is applied on top of TMC13, their qp ranges are identical

The semantic information is treated as a point attribute in the compression implementations (attribute coding). This was far more efficient (in terms of rate) than other coding mechanisms (e.g., arithmetic coding), as they are unaware of the properties of PCs.

4.1. Experiments

The results reported in this section have been obtained by averaging compression results on the first 100 point clouds of the SemanticKITTI validation set (sequence 08). Our method is tested on Draco and TMC13, and the coding performance is evaluated through a comparison with the corresponding standard codecs in terms of PSNR D1 (Figure 3) and execution time (Figure 4).

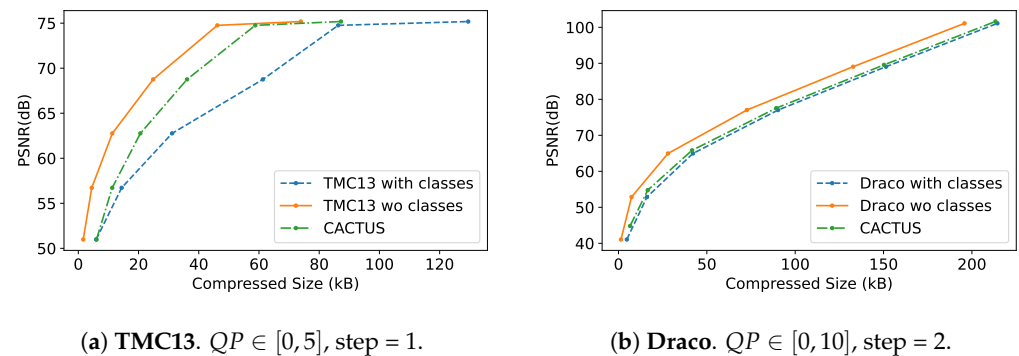


Figure 3. Comparison between TMC13 and Draco rate curves on the full point cloud, with geometry only (orange), with geometry and semantic information (blue), and with CACTUS (green).

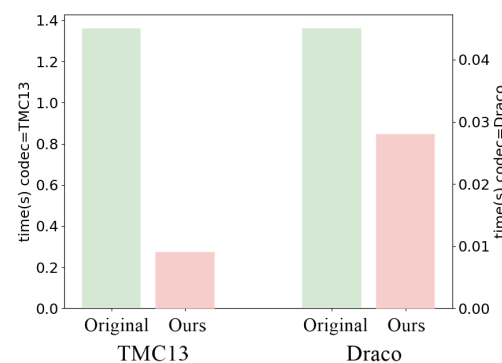


Figure 4. Execution times for CACTUS and the reference codecs.

4.2. Results on TMC13 and Draco

Instead, the accuracy of segmentation is measured through the mean Intersection over Union (mIoU), which reports how close the obtained segmentation is with respect to

some ground-truth data. Since the main problem tackled by the CACTUS framework is the transmission of point cloud geometry while preserving its semantic characterization, some preliminary tests concerning the impact of coding distortion on the segmentation accuracy need to be performed. This analysis is required because most of the SS architectures are trained and tested on clean data (not corrupted by quantization noise). Therefore, in a realistic scenario where compression of the signal is essential, the performance of the SS algorithm on compressed and reconstructed point clouds might be strongly impaired. To the best of our knowledge, compression effects on LiDAR semantic segmentation are objectively analyzed here for the first time.

Before carrying on, it is worth reminding that whenever compression follows semantic segmentation, some points, which were originally distinct, are overlapping because of the quantization of spatial coordinates. Note that in this case, they could have been assigned to different semantic classes (this is likely to happen in regions of transition between different instances), making the computation of the final mIoU very hard. The impact of compression on segmentation is therefore made using only the TMC13 codec, since Draco does not remove overlapping points.

The decimation on the reconstructed point cloud and the quantization noise that affects the spatial coordinates severely impair the accuracy of object classification and localization. This is evaluated by comparing the mIoU between the ground truth and the segmented data before and after the compression. This procedure generates $\text{mIoU}(y_{net}^{qp}, y_{gt}^{qp})$ and $\text{mIoU}(y_{codec}^{qp}, y_{gt}^{qp})$. Note that:

- y_{net}^{qp} is the output predicted by the NN by processing the geometry of a PC compressed with factor qp .
- y_{codec}^{qp} are the labels obtained by assigning to each compressed point the label predicted by RandLA-Net for its nearest neighbor in the original geometry (recoloring in TMC13).
- y_{gt}^{qp} is the PC recolored using the original ground truth instead of the RandLA-Net prediction.

Figure 5 displays the difference $\Delta \text{mIoU} = \text{mIoU}(y_{codec}^{qp}, y_{gt}^{qp}) - \text{mIoU}(y_{net}^{qp}, y_{gt}^{qp})$ versus the QP qp . It is possible to notice that even with $qp = 0$, the network has a very big loss in performance mostly due to a rounding of the coordinate values to the chosen unit of measurement (sensor precision). Indeed, even very small quantization noise can affect the segmentation accuracy at the receiver mounting up for higher qp values. This effect was also verified by adding random uniform noise to point coordinates (interval $[-50 \text{ mm}, 50 \text{ mm}]$), leading to a comparable loss and showing that the architecture is very susceptible to noise.

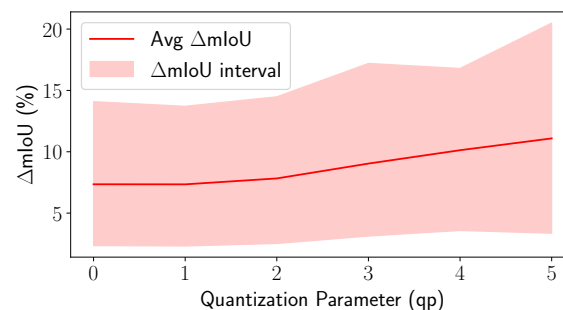


Figure 5. Difference between $\text{mIoU}(y_{codec}^{qp}, y_{gt}^{qp})$ and $\text{mIoU}(y_{net}^{qp}, y_{gt}^{qp})$.

By considering the execution time (omitting the SS step, since it has to be carried out regardless) experimental results (see Figure 4) show that CACTUS is approximately five times faster than TMC13 and two times faster than Draco at encoding and decoding geometry and classes. This is supported by the fact that in general, splitting a problem

that can be solved with an algorithm with super-linear complexity (PCC) into independent subproblems (encoding subsets of the PC with the same semantic class) should reduce the overall number of required operations (see Appendix A).

In the case of PCC, the compression algorithms are assumed to have at least $\mathcal{O}(n)$ complexity. In fact, they need to process each point at least once, but it appears to be super-linear (e.g., in order to remove duplicate points, TMC13 performs a $\mathcal{O}(n \log n)$ sorting operation). So, there are the conditions to assume that splitting the LiDAR signal in c classes actually lowers the upper bound to the total number of operations in the coding procedure. This is not guaranteed to reduce the execution time, but it appears to do so with high probability, since empirical results show that on average CACTUS reduces the encoding time. On top of that, the CACTUS workflow is highly parallelizable, as different parts can be independently encoded, leading to a lower execution time when multiple cores are available. This analysis supports the claim that the proposed framework proves to be efficient with respect to state-of-the-art solutions.

The rate-distortion performance of the codec is analyzed using the quality metrics described in [61]. The plots reported in Figure 3 display the Peak Signal to Noise Ratio (PSNR) D1 (measured in dB) of the reconstructed PC versus the size in kB of the coded stream. In our tests, only PSNR D1 is used for quality evaluation, since PSNR D2 implies the estimation of normals which can not be easily performed on sparse point clouds. Indeed, the latter metrics is mainly used for dense point sets.

As expected, coding semantic classes results in higher rates at constant geometry quality. This is due to the addition of attribute information when encoding with TMC13 and Draco in the case of CACTUS, because splitting the PC into subsets makes it harder to fully exploit the spatial redundancy for the encoder. However, whenever semantic information is required by the receiver (or a more precise rate control needs to be entailed), using a CACTUS segmentation-based codec proves to be more efficient, especially in the case of TMC13.

This result holds for all the samples and the Bjontegaard delta rate computed between the curves produced by CACTUS and TMC13 with class attributes are on average $\Delta rate_i = -31.16\%$, $\Delta PSNR_i = 3.52$. Conversely, it does not always hold for Draco where segmentation generally does not yield significant rate reductions. However, on average, CACTUS is superior in execution time, Rate Distortion (RD) trade-off and rate allocation flexibility, reading $\Delta rate_i = -5.26\%$, $\Delta PSNR_i = 0.98$.

Additionally, encoding the whole PC usually results in higher errors in under-represented classes (since a higher percentage of their points is removed because of quantization): for this reason, we also decided to evaluate performance in terms of average per-class PSNR

$$PSNR_{avg}(P, P^{comp}) = \frac{1}{|c(P)|} \sum_{i \in c(P)} PSNR(P_i, P_i^{comp}) \quad (2)$$

where $c(P)$ is the set of classes that can be found in the original point cloud, and P_i, P_i^{comp} are the subsets of points with class c_i from P and P^{comp} , respectively, where the latter is the compressed version of the former. Using this distortion metric, it is possible to obtain the plots in Figure 6a,b (results obtained for TMC13 and Draco are relative to the bitrates obtained without encoding classes). In general, CACTUS achieves better RD performance than its counterparts, even if they have a major advantage in terms of rate, since they do not require encoding classes. This shows that this type of approach is better at preserving the finer details of the less populated classes that do not contribute as much to the total PSNR.

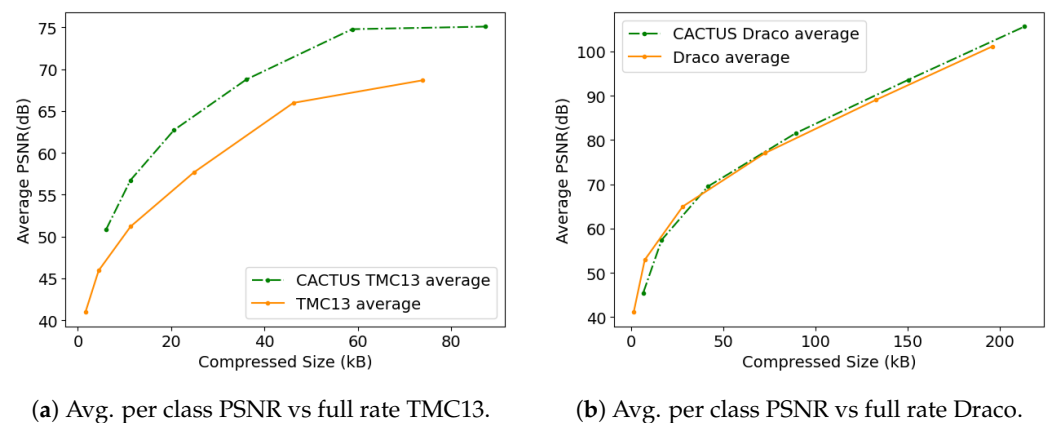


Figure 6. Comparison between the average per class PSNR computed with the standard codecs and with CACTUS.

4.3. Results with Neural Codecs

As mentioned above, we have tested the CACTUS framework with [24] because thanks to side information, it is more robust to the shift in distribution due to the difference in the statistics of the training data and the LiDAR data. However, most of the results and the considerations that will be drawn later actually apply to most LPCGCs, since they often process PCs in the voxel domain.

In Figure 7, it is possible to see that similarly to the standard codecs case, also that with learned ones CACTUS results in slightly lower RD performance if the receiver is only interested in the geometry. However, as before when semantic information is required, then CACTUS proves to be much more effective than using standard attribute compression algorithms such as RAHT.

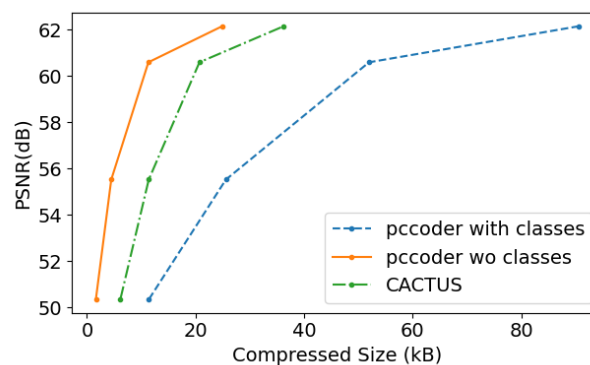


Figure 7. Comparison between rate curves on the full point cloud with geometry only (DSAE), with geometry and semantic information (DSAE + RAHT), and with CACTUS.

Even though CACTUS shows similar behaviors in terms of RD performance even when using learned codecs, the same cannot be said when considering the computational cost. As a matter of fact, neural networks are inherently parallelized on specialized hardware, making it harder to assess the actual difference in encoding and decoding time. However, assuming sequential operations, i.e., using a single core on a CPU, CACTUS would result in a higher number of total operations due to the usage of voxelized representation.

As a matter of fact, while splitting the data based on the semantic class does not change the overall number of inputs when using a point-based representation, the same does not hold with voxels, since partitioning the total number of cells in input to the codec can change in different ways. Consider for example a PC with n classes; then, the total number of voxels in input to the codec can:

- Increase: e.g., if the class of each point is assigned at random, then the total number of voxels is going to increase by a factor of n after partitioning.
- Remain constant: if the PC can be perfectly divided in n regions with no overlap, then the total number of voxels will not change.
- Decrease: there are some configurations, although rare, where there are empty regions in the PC that can be ignored after partitioning because of a better estimation of the bounds of the PC, which can lead to a lower number of voxels being fed to the codec.

This makes it much harder to analyze the complexity of the approach when using neural networks. However, in the sequential case, the execution time is likely going to increase with respect to encoding the whole PC in one pass, since perfect partitioning with no region overlap is very unlikely in this scenario. However, when considering parallel computing capabilities, this issue should be considerably mitigated.

5. Discussion

From the aforementioned results, it is possible to point out most of the main advantages and the main disadvantages of the proposed approach. In particular, the main strengths of the CACTUS framework are:

1. Across all the proposed codecs (see Figures 3a,b and 7), when semantic information needs to be transmitted, it is more effective to use CACTUS than the default codecs. This is likely due to the fact that standard attribute coding procedures are designed for continuous values (such as color and normals), and they are not suited for categorical labels, leading to sub-optimal performance. This is solved by CACTUS, which independently encodes the geometry of different semantic classes, allowing the transmission of this type of information almost for free.
2. From the analysis in Appendix A and the empirical results in Figure 4, it is possible to see that when the points are encoded without changing their representation, and when the algorithm complexity is super-linear (which is almost always the case), then independently compressing partitions of the PC leads to lower encoding and decoding time. This is especially useful in the automotive scenario where working in real time is a very important requirement. This does not always apply for neural codecs, since most of them process voxelized representations. Additionally, since in those cases parallelization is often exploited, the execution time highly depends on the specific PC that is being processed and on the classes distribution.
3. As can be seen from Figure 5, independently transmitting semantically disjoint partitions helps improve the quality of the class information at the receiver side for various reasons. Firstly, encoding all the PC with a standard codec can lead to the removal of duplicate points, which is non-optimal, since important information (such as the presence of a pedestrian) could be removed from the scene, and secondly, the performance of current SS strategies are hindered by compression noise, as shown by the analysis carried out in this paper.
4. This type of approach has much greater flexibility than simply using a standard codec, since it allows developing rate allocation algorithms that can choose different coding parameters for each semantic class depending on the requirements of the application. There are, however, some drawbacks such as:

1. There are some use cases where CACTUS is sub-optimal. In particular, if all points are equally important for the end task and if the receiver does not care about semantic information, then using the base codec allows achieving better RD performance. This can be easily seen from Figures 3a,b and 7, where the geometry RD curves of TMC13, Draco and DSAE are above the corresponding CACTUS ones. The main reason for this is that the latter further sparsifies the PC, making it hard to fully exploit the spatial correlation in the data.

2. Many LPCGCs struggle when reconstructing sparse PCs, and the issue is further amplified by CACTUS. This makes it non-trivial to seamlessly use them in the framework.

6. Conclusions and Future Works

This paper reports a detailed investigation of the problems and the possibilities derived by the combination of semantic segmentation with point cloud coding in a collaborative automotive scenario. Semantic segmentation strategies have proved to be highly susceptible to quantization noise, thus resulting in sub-optimal performance when applied at the receiver side. The current paper also proposes a new coding framework for LiDAR PCs that processes and transmits semantic classes independently, leading to higher flexibility in shaping the bit stream. Simultaneously, the coding time is reduced, reliable semantic data can be provided at the receiver, and a more flexible bitrate allocation is enabled, allowing modulating the reconstruction quality on the semantic class. The transmission of class information proves to be efficient also in rate-distortion terms, since CACTUS RD performance shows a negligible loss in modeling geometry and a compression gain in coding semantic information. Such advantages have been investigated on three different coding schemes; however, the approach is totally generalizable to other codecs or segmentation strategies.

Therefore, if high flexibility is required or some semantic information needs to be transmitted, the proposed framework provides improvements in terms of RD performance, execution time, and mIoU at the receiver. Future research should focus on improving the framework by designing the codec and possibly the SS algorithm to better interact together, and by developing algorithms that are more robust to compression noise, in particular:

- The codecs that are currently proposed in the literature do not use semantic information when performing compression; therefore, the performance of CACTUS might be considerably improved with an ad hoc codec that is conditioned on the class of the considered partition. One other possibility would be to make the codec autoregressive, i.e., the decoder could exploit the already decoded parts of the PC to infer some information about the parts that it still has not reconstructed. This usually leads to bitrate savings which could mitigate the fact that independently decoding the partitions makes it harder for the system to exploit spatial redundancy.
- The effect of other SS schemes on the framework should be explored to assess their effect on the overall coding performance.
- Furthermore, this work shows that compression noise can considerably hinder the performance of scene understanding algorithms. This should strongly motivate researchers to study some new techniques to make their models more robust to this and other types of noises. For example, training the NNs on compressed data might make them much more robust to the modified data distribution.
- It might be possible to train the whole system (semantic segmentation model and codec) in an end-to-end fashion to achieve better representational capabilities and faster inference due to the features shared by the two blocks.

Author Contributions: Conceptualization, D.M., E.C. and S.M.; methodology, D.M.; software, D.M.; validation, D.M.; formal analysis, D.M., E.C. and S.M.; investigation, D.M.; resources, S.M.; data elaboration, D.M. and E.C.; writing—original draft preparation, D.M., E.C. and S.M.; writing—review and editing, D.M., E.C. and S.M.; visualization, D.M. and E.C.; supervision, S.M.; project administration, S.M.; funding acquisition, S.M. All authors have read and agreed to the published version of the manuscript.

Funding: SID 2018 project “SartreMR”—Department of Information Engineering, University of Padova. This work was partially supported by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on “Telecommunications of the Future” (PE00000001—program “RESTART”). Daniele Mari’s activities were supported by Fondazione CaRiPaRo under the grants “Dottorati di Ricerca” 2021/2022.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All of the datasets mentioned in this paper are publicly available and properly cited in bibliography. These data can be found here: <http://www.semantic-kitti.org/dataset.html> accessed date: 15 January 2023.

Acknowledgments: We acknowledge all the funders reported in the previous sections.

Conflicts of Interest: The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

CACTUS	Content-Aware Compression and Transmission Using Semantics
DL	Deep Learning
DSAE	Distributed Source Auto-Encoder
GPCC	Geometry-based Point Cloud Coding
LiDAR	Light Detection and Ranging
LPCGC	Learned Point Cloud Geometry Codec
mIoU	mean Intersection over Union
NN	Neural Network
PC	Point Cloud
PCC	Point Cloud Coding
PSNR	Peak Signal-to-Noise Ratio
QP	Quantization Parameter
RAHT	Region Adaptive Hierarchical Transform
RD	Rate Distortion
SS	Semantic Segmentation

Appendix A

Suppose that we have an algorithm that has computational complexity $\Theta(n^\gamma(\log n)^\delta)$ where then this means that:

$$\exists k_1, k_2, n_0 : c_{k_1}(n) = k_1 n^\gamma (\log n)^\delta < a(n) < k_2 n^\gamma (\log n)^\delta = c_{k_2}(n) \quad \forall n > n_0 \quad (\text{A1})$$

where $a(n)$ is the actual number of operations performed by the algorithm, $k, \gamma, \delta \in \mathbb{R} : k > 0, \gamma > 1, \delta > 0$.

If it is possible, for some reason, to process the input by independently running the algorithm $m \in \{2, 3, \dots, n\}$ times on subsets of the input of size $\frac{n}{m}$; then, the bound for a general value k becomes:

$$\sum_{i=1}^m c_k\left(\frac{n}{m}\right) = m c_k\left(\frac{n}{m}\right) = m k \left(\frac{n}{m}\right)^\gamma \left(\log \frac{n}{m}\right)^\delta \quad (\text{A2})$$

However, it is simple to see by manipulating Equation (A2) that $m c_k\left(\frac{n}{m}\right) < c_k(n)$ since

$$\begin{aligned} m c_k\left(\frac{n}{m}\right) &= m k \left(\frac{n}{m}\right)^\gamma \left(\log \frac{n}{m}\right)^\delta \\ &= k \frac{n^\gamma}{m^{\gamma-1}} \left(\log n - \log m\right)^\delta \\ &< k \frac{n^\gamma}{m^{\gamma-1}} \log^\delta n \\ &< k n^\gamma \log^\delta n = c_k(n) \end{aligned} \quad (\text{A3})$$

where the first inequality comes from the fact that $1 < m \leq n, \delta > 1$, while the second one is due to the fact that since $\gamma > 1, m > 1$, then $m^{\gamma-1} > 1$, i.e., $\frac{1}{m^{\gamma-1}} < 1$. Similarly, it is possible to obtain a similar result even if the dataset is not equally split in m folds; however, the multiplicative constant in the bounds is reduced by a smaller amount.

For this reason, it is possible to see that the CACTUS coding complexity given the semantic labels is asymptotically bounded by two equations with smaller multiplication constants $\frac{k_1}{m^{\gamma-1}} < k_1$, $\frac{k_2}{m^{\gamma-1}} < k_2$ with respect to the respective ones in the standard compression case. This always leads to a smaller number of operations for $n > n_0$ if $\frac{k_2}{m^{\gamma-1}} < k_1$; otherwise, it will generally do so but not with mathematical certainty. However, empirical observations confirmed that this is the case for CACTUS.

References

1. Prokop, M.; Shaikh, S.A.; Kim, K. Low Overlapping Point Cloud Registration Using Line Features Detection. *Remote Sens.* **2020**, *12*, 61. [\[CrossRef\]](#)
2. Mongus, D.; Brumen, M.; Zlaus, D.; Kohek, S.; Tomazic, R.; Kerin, U.; Kolmanic, S. A Complete Environmental Intelligence System for LiDAR-Based Vegetation Management in Power-Line Corridors. *Remote Sens.* **2021**, *13*, 5159. [\[CrossRef\]](#)
3. Arastounia, M. Automated As-Built Model Generation of Subway Tunnels from Mobile LiDAR Data. *Sensors* **2016**, *16*, 1486. [\[CrossRef\]](#) [\[PubMed\]](#)
4. Camuffo, E.; Mari, D.; Milani, S. Recent Advancements in Learning Algorithms for Point Clouds: An Updated Overview. *Sensors* **2022**, *22*, 1357. [\[CrossRef\]](#)
5. Meijer, C.; Grootes, M.; Koma, Z.; Dzigan, Y.; Gonçalves, R.; Andela, B.; van den Oord, G.; Rangelova, E.; Renaud, N.; Kissling, W. Laserchicken—A tool for distributed feature calculation from massive LiDAR point cloud datasets. *SoftwareX* **2020**, *12*, 100626. [\[CrossRef\]](#)
6. Marvasti, E.E.; Raftari, A.; Marvasti, A.E.; Fallah, Y.P.; Guo, R.; Lu, H. Cooperative LIDAR Object Detection via Feature Sharing in Deep Networks. In Proceedings of the 2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall), Virtual, 18 November–16 December 2020; pp. 1–7. [\[CrossRef\]](#)
7. Dimitrievski, M.; Jacobs, L.; Veelaert, P.; Philips, W. People Tracking by Cooperative Fusion of RADAR and Camera Sensors. In Proceedings of the 2019 IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, New Zealand, 27–30 October 2019; pp. 509–514. [\[CrossRef\]](#)
8. Guan, H.; Li, J.; Zhong, L.; Yongtao, Y.; Chapman, M. Process virtualization of large-scale lidar data in a cloud computing environment. *Comput. Geosci.* **2013**, *60*, 109–116. [\[CrossRef\]](#)
9. Hegeman, J.W.; Sardeshmukh, V.B.; Sugumaran, R.; Armstrong, M.P. Distributed LiDAR data processing in a high-memory cloud-computing environment. *Ann. GIS* **2014**, *20*, 255–264. [\[CrossRef\]](#)
10. Sarker, V.K.; Peña Queralta, J.; Gia, T.N.; Tenhunen, H.; Westerlund, T. Offloading SLAM for Indoor Mobile Robots with Edge-Fog-Cloud Computing. In Proceedings of the 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), Dhaka, Bangladesh, 3–5 May 2019; pp. 1–6. [\[CrossRef\]](#)
11. Shin, S.; Kim, J.; Moon, C. Road Dynamic Object Mapping System Based on Edge-Fog-Cloud Computing. *Electronics* **2021**, *10*, 2825. [\[CrossRef\]](#)
12. Graziosi, D.; Nakagami, O.; Kuma, S.; Zaghetto, A.; Suzuki, T.; Tabatabai, A. An overview of ongoing point cloud compression standardization activities: Video-based (V-PCC) and geometry-based (G-PCC). *APSIPA Trans. Signal Inf. Process.* **2020**, *9*, e13. [\[CrossRef\]](#)
13. Google. Draco 3D Data Compression. 2017. Available online: <https://github.com/google/draco> (accessed on 15 January 2023).
14. Huang, L.; Wang, S.; Wong, K.; Liu, J.; Urtasun, R. Octsqueezer: Octree-structured entropy model for lidar compression. In Proceedings of the CVPR 2020, Seattle, WA, USA, 13–19 June 2020; pp. 1313–1323.
15. Meagher, D. Geometric modeling using octree encoding. *Comput. Graph. Image Process.* **1982**, *19*, 129–147. [\[CrossRef\]](#)
16. Tu, C.; Takeuchi, E.; Carballo, A.; Takeda, K. Point cloud compression for 3D LiDAR sensor using recurrent neural network with residual blocks. In Proceedings of the ICRA 2019, Montreal, QC, Canada, 20–24 May 2019; pp. 3274–3280.
17. Varischio, A.; Mandruzzato, F.; Bullo, M.; Giordani, M.; Testolina, P.; Zorzi, M. Hybrid Point Cloud Semantic Compression for Automotive Sensors: A Performance Evaluation. *arXiv* **2021**, arXiv:2103.03819.
18. Guarda, A.F.R.; Rodrigues, N.M.M.; Pereira, F. Point Cloud Coding: Adopting a Deep Learning-based Approach. In Proceedings of the 2019 Picture Coding Symposium (PCS), Ningbo, China, 12–15 November 2019; pp. 1–5. [\[CrossRef\]](#)
19. Guarda, A.F.; Rodrigues, N.M.; Pereira, F. Adaptive deep learning-based point cloud geometry coding. *IEEE J. Sel. Top. Signal Process.* **2020**, *15*, 415–430. [\[CrossRef\]](#)
20. Guarda, A.F.; Rodrigues, N.M.; Pereira, F. Point cloud geometry scalable coding with a single end-to-end deep learning model. In Proceedings of the 2020 IEEE International Conference on Image Processing (ICIP), Anchorage, AK, USA, 19–22 September 2020; pp. 3354–3358.
21. Quach, M.; Valenzise, G.; Dufaux, F. Learning convolutional transforms for lossy point cloud geometry compression. In Proceedings of the 2019 IEEE International Conference on Image Processing (ICIP), Anchorage, AK, USA, 19–22 September 2020; pp. 4320–4324.
22. Quach, M.; Valenzise, G.; Dufaux, F. Improved deep point cloud geometry compression. In Proceedings of the 2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP), Virtual, 21–24 September 2020; pp. 1–6.

23. Milani, S. ADAE: Adversarial Distributed Source Autoencoder For Point Cloud Compression. In Proceedings of the IEEE ICIP 2021, Bordeaux, France, 16–19 October 2021; pp. 3078–3082.
24. Milani, S. A Syndrome-Based Autoencoder For Point Cloud Geometry Compression. In Proceedings of the IEEE ICIP 2020, Abu Dhabi, United Arab Emirates, 25–28 October 2020; pp. 2686–2690. [\[CrossRef\]](#)
25. Milani, S.; Calvagno, G. A cognitive approach for effective coding and transmission of 3D video. *ACM Trans. Multimed. Comput. Commun. Appl. (TOMCCAP)* **2011**, *75*, 1–21. [\[CrossRef\]](#)
26. Hu, Q.; Yang, B.; Xie, L.; Rosa, S.; Guo, Y.; Wang, Z.; Trigoni, N.; Markham, A. Randla-net: Efficient semantic segmentation of large-scale point clouds. In Proceedings of the CVPR 2020, Seattle, WA, USA, 13–19 June 2020; pp. 11108–11117.
27. Geiger, A.; Lenz, P.; Urtasun, R. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In Proceedings of the CVPR 2012, Providence, RI, USA, 16–21 June 2012; pp. 3354–3361.
28. Behley, J.; Garbade, M.; Milioto, A.; Quenzel, J.; Behnke, S.; Stachniss, C.; Gall, J. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In Proceedings of the ICCV 2019, Seoul, Republic of Korea, 27–28 October 2019.
29. St. Peter, J.; Drake, J.; Medley, P.; Ibeanusi, V. Forest Structural Estimates Derived Using a Practical, Open-Source Lidar-Processing Workflow. *Remote Sens.* **2021**, *13*, 4763. [\[CrossRef\]](#)
30. Cabo, C.; Ordóñez, C.; Lasheras, F.S.; Roca-Pardiñas, J.; de Cos Juez, F.J. Multiscale Supervised Classification of Point Clouds with Urban and Forest Applications. *Sensors* **2019**, *19*, 4523. [\[CrossRef\]](#)
31. Cortés Gallardo Medina, E.; Velazquez Espitia, V.M.; Chípuli Silva, D.; Fernández Ruiz de las Cuevas, S.; Palacios Hirata, M.; Zhu Chen, A.; González González, J.Á.; Bustamante-Bello, R.; Moreno-García, C.F. Object detection, distributed cloud computing and parallelization techniques for autonomous driving systems. *Appl. Sci.* **2021**, *11*, 2925. [\[CrossRef\]](#)
32. Arthurs, P.; Gillam, L.; Krause, P.; Wang, N.; Halder, K.; Mouzakitis, A. A Taxonomy and Survey of Edge Cloud Computing for Intelligent Transportation Systems and Connected Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 6206–6221. [\[CrossRef\]](#)
33. Lee, J.; Lee, K.; Yoo, A.; Moon, C. Design and Implementation of Edge-Fog-Cloud System through HD Map Generation from LiDAR Data of Autonomous Vehicles. *Electronics* **2020**, *9*, 2084. [\[CrossRef\]](#)
34. Nevalainen, P.; Li, Q.; Melkas, T.; Riekk, K.; Westerlund, T.; Heikkonen, J. Navigation and Mapping in Forest Environment Using Sparse Point Clouds. *Remote Sens.* **2020**, *12*, 4088. [\[CrossRef\]](#)
35. Miller, I.D.; Cowley, A.; Konkimalla, R.; Shivakumar, S.S.; Nguyen, T.; Smith, T.; Taylor, C.J.; Kumar, V. Any Way You Look at It: Semantic Crossview Localization and Mapping With LiDAR. *IEEE Robot. Autom. Lett.* **2021**, *6*, 2397–2404. [\[CrossRef\]](#)
36. Huang, Y.; Shan, T.; Chen, F.; Englot, B. DiSCo-SLAM: Distributed Scan Context-Enabled Multi-Robot LiDAR SLAM with Two-Stage Global-Local Graph Optimization. *IEEE Robot. Autom. Lett.* **2021**, *36*, 1150–1157. [\[CrossRef\]](#)
37. Sualeh, M.; Kim, G.W. Dynamic Multi-LiDAR Based Multiple Object Detection and Tracking. *Sensors* **2019**, *19*, 1474. [\[CrossRef\]](#)
38. Massa, F.; Bonamini, L.; Settimi, A.; Pallottino, L.; Caporale, D. LiDAR-Based GNSS Denied Localization for Autonomous Racing Cars. *Sensors* **2020**, *20*, 3992. [\[CrossRef\]](#)
39. Shan, M.; Narula, K.; Wong, Y.F.; Worrall, S.; Khan, M.; Alexander, P.; Nebot, E. Demonstrations of Cooperative Perception: Safety and Robustness in Connected and Automated Vehicle Operations. *Sensors* **2021**, *21*, 200. [\[CrossRef\]](#)
40. Camuffo, E.; Gorghetto, L.; Badia, L. Moving Drones for Wireless Coverage in a Three-Dimensional Grid Analyzed via Game Theory. In Proceedings of the 2021 IEEE Asia Pacific Conference on Circuit and Systems (APCCAS), Penang, Malaysia, 22–26 November 2021; pp. 41–44.
41. Tang, S.; Chen, B.; Iwen, H.; Hirsch, J.; Fu, S.; Yang, Q.; Palacharla, P.; Wang, N.; Wang, X.; Shi, W. VECFrame: A Vehicular Edge Computing Framework for Connected Autonomous Vehicles. In Proceedings of the 2021 IEEE International Conference on Edge Computing (EDGE), Chicago, IL, USA, 5–10 September 2021; pp. 68–77. [\[CrossRef\]](#)
42. Chen, W.; Zhou, S.; Pan, Z.; Zheng, H.; Liu, Y. Mapless Collaborative Navigation for a Multi-Robot System Based on the Deep Reinforcement Learning. *Appl. Sci.* **2019**, *9*, 4198. [\[CrossRef\]](#)
43. Zhou, L.; Geng, J.; Jiang, W. Joint Classification of Hyperspectral and LiDAR Data Based on Position-Channel Cooperative Attention Network. *Remote Sens.* **2022**, *14*, 3247. [\[CrossRef\]](#)
44. Zhang, Z.; Li, T.; Tang, X.; Lei, X.; Peng, Y. Introducing Improved Transformer to Land Cover Classification Using Multispectral LiDAR Point Clouds. *Remote Sens.* **2022**, *14*, 3808. [\[CrossRef\]](#)
45. Chen, L.; Fan, X.; Jin, H.; Sun, X.; Cheng, M.; Wang, C. FedRME: Federated Road Markings Extraction from Mobile LiDAR Point Clouds. In Proceedings of the 2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD), Hangzhou, China, 4–6 May 2022; pp. 653–658. [\[CrossRef\]](#)
46. Thanou, D.; Chou, P.A.; Frossard, P. Graph-based compression of dynamic 3D point cloud sequences. *IEEE Trans. Image Process.* **2016**, *25*, 1765–1778. [\[CrossRef\]](#)
47. de Oliveira Rente, P.; Brites, C.; Ascenso, J.; Pereira, F. Graph-based static 3D point clouds geometry coding. *IEEE Trans. Multimed.* **2018**, *21*, 284–299. [\[CrossRef\]](#)
48. Wang, J.; Zhu, H.; Ma, Z.; Chen, T.; Liu, H.; Shen, Q. Learned point cloud geometry compression. *arXiv* **2019**, arXiv:1909.12037.
49. Achlioptas, P.; Diamanti, O.; Mitliagkas, I.; Guibas, L. Learning representations and generative models for 3d point clouds. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 40–49.
50. Yang, Y.; Feng, C.; Shen, Y.; Tian, D. Foldingnet: Point cloud auto-encoder via deep grid deformation. In Proceedings of the CVPR 2018, Salt Lake City, UT, USA, 18–22 June 2018; pp. 206–215.

51. Devillers, O.; Gandoin, P.M. Geometric compression for interactive transmission. In Proceedings of the VIS 2000 (Cat. No. 00CH37145), Salt Lake City, UT, USA, 8–13 October 2000; pp. 319–326.
52. Milani, S.; Polo, E.; Limuti, S. A Transform Coding Strategy for Dynamic Point Clouds. *IEEE Trans. Image Process.* **2020**, *29*, 8213–8225. [[CrossRef](#)]
53. Zhao, L.; Ma, K.K.; Lin, X.; Wang, W.; Chen, J. Real-Time LiDAR Point Cloud Compression Using Bi-Directional Prediction and Range-Adaptive Floating-Point Coding. *IEEE Trans. Broadcast.* **2022**, *68*, 620–635. [[CrossRef](#)]
54. Sun, X.; Wang, S.; Liu, M. A Novel Coding Architecture for Multi-Line LiDAR Point Clouds Based on Clustering and Convolutional LSTM Network. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 2190–2201. [[CrossRef](#)]
55. Milioto, A.; Vizzo, I.; Behley, J.; Stachniss, C. Rangenet++: Fast and accurate lidar semantic segmentation. In Proceedings of the IROS 2019, Macau, China, 3–8 November 2019; pp. 4213–4220.
56. Matrone, F.; Grilli, E.; Martini, M.; Paolanti, M.; Pierdicca, R.; Remondino, F. Comparing Machine and Deep Learning Methods for Large 3D Heritage Semantic Segmentation. *ISPRS Int. J. Geo-Inf.* **2020**, *9*, 535. [[CrossRef](#)]
57. Zhou, H.; Zhu, X.; Song, X.; Ma, Y.; Wang, Z.; Li, H.; Lin, D. Cylinder3D: An Effective 3D Framework for Driving-scene LiDAR Semantic Segmentation. *arXiv* **2020**, arXiv:2008.01550.
58. De Queiroz, R.L.; Chou, P.A. Compression of 3D point clouds using a region-adaptive hierarchical transform. *IEEE Trans. Image Process.* **2016**, *25*, 3947–3956. [[CrossRef](#)]
59. Pradhan, S.S.; Ramchandran, K. Distributed source coding using syndromes (DISCUS): Design and construction. *IEEE Trans. Inf. Theory* **2003**, *49*, 626–643. [[CrossRef](#)]
60. Wu, Z.; Song, S.; Khosla, A.; Yu, F.; Zhang, L.; Tang, X.; Xiao, J. 3d shapenets: A deep representation for volumetric shapes. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 1912–1920.
61. MPEG 3DG; Requirements. Common test conditions for point cloud compression; document N17229. In Proceedings of the JCT_MEET Meeting Proceedings, 2017.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.