sensors

MDPI

Article

An Efficient LiDAR Point Cloud Map Coding Scheme Based on Segmentation and Frame-Inserting Network

Qiang Wang [1,2], Liuyang Jiang [1,3], Xuebin Sun [4], Jingbo Zhao [1], Zhaopeng Deng [1] and Shizhong Yang [1,*]

[1] College of Information and Control Engineering, Qingdao University of Technology, Qingdao 266525, China; wangqiang@qut.edu.cn (Q.W.); jiangliuyang@qut.edu.cn (L.J.); zhaojbqut@163.com (J.Z.); dengzhaopeng@qut.edu.cn (Z.D.)
[2] State Key Laboratory of Precision Measuring Technology and Instruments, Tianjin University, Tianjin 300072, China
[3] School of Automation Science and Electrical Engineering, Beihang University, Beijing 100083, China
[4] School of Electronics and Information Engineering, Shenzhen University, Shenzhen 518060, China; sunxuebin@szu.edu.cn
* Correspondence: yangszqut@163.com; Tel.: +86-0532-8687-2105

Abstract: In this article, we present an efficient coding scheme for LiDAR point cloud maps. As a point cloud map consists of numerous single scans spliced together, by recording the time stamp and quaternion matrix of each scan during map building, we cast the point cloud map compression into the point cloud sequence compression problem. The coding architecture includes two techniques: intra-coding and inter-coding. For intra-frames, a segmentation-based intra-prediction technique is developed. For inter-frames, an interpolation-based inter-frame coding network is explored to remove temporal redundancy by generating virtual point clouds based on the decoded frames. We only need to code the difference between the original LiDAR data and the intra/inter-predicted point cloud data. The point cloud map can be reconstructed according to the decoded point cloud sequence and quaternion matrices. Experiments on the KITTI dataset show that the proposed coding scheme can largely eliminate the temporal and spatial redundancies. The point cloud map can be encoded to 1/24 of its original size with 2 mm-level precision. Our algorithm also obtains better coding performance compared with the octree and Google Draco algorithms.

Keywords: LiDAR; point cloud map; coding; segmentation; interpolation

Citation: Wang, Q.; Jiang, L.; Sun, X.; Zhao, J.; Deng, Z.; Yang, S. An Efficient LiDAR Point Cloud Map Coding Scheme Based on Segmentation and Frame-Inserting Network. *Sensors* **2022**, *22*, 5108. https://doi.org/10.3390/s22145108

Academic Editors: Baochang Zhang and Ying Huang

Received: 6 June 2022
Accepted: 6 July 2022
Published: 7 July 2022

1. Introduction

LiDAR point clouds have been widely used in many emerging applications [1], such as the preservation of historical relics, mobile robots, and remote sensing [2–9]. LiDAR sensors are essential for autonomous vehicles, and dense LiDAR point cloud maps play an indispensable role in unmanned driving, such as obstacle detection [10], localization [11], and navigation [12]. In wide geographic areas, a LiDAR point cloud map consists of a vast number of points and requires a large bandwidth and storage space to transmit and store [4,13–15]. Therefore, developing compression algorithms for the dense LiDAR point cloud maps is an urgent task.

With the characteristics of covering a large area, unstructured organization, and a huge volume [16], it is difficult to remove redundancy without distortion when encoding the LiDAR point cloud map. Octree, as a data structure, has been widely used to encode point clouds. Each internal node has exactly eight children in an octree. The octree-based point cloud compression method is to divide a 3D point cloud by recursively subdividing it into eight octants. As octree-based compression methods are lossy, they are suboptimal for the autonomous vehicles that have strict requirements for compression accuracy in the task of path planning or obstacle detection, etc.

In this research, we focus on compressing the large-scale dense LiDAR point cloud maps. The well-known low-drift and real-time LiDAR odometry (LOAM) algorithm are used to build the 3D map [17]. The proposed LiDAR point cloud map coding algorithm can be used in mobile robots or surveying and mapping fields. The major contributions are as follows.

- By recording the time stamp and quaternion matrix of each scan during mapping, the large-scale point cloud map compression can be formulated as a point cloud sequence compression problem;
- For intra-coding, we develop an intra-prediction method based on segmentation and plane fitting, which can exploit and remove the spatial redundancy by utilizing the spatial structure characteristics of the point cloud.
- For inter-coding, we develop an interpolation-based inter-prediction network, in which the previous time and the next time encoded point clouds are utilized to synthesize the point clouds of the intermediate time to remove the temporal redundancy.
- Experimental results on the KITTI dataset demonstrate that the proposed method achieves a competitive compression performance for the dense LiDAR point cloud maps compared with other state-of-the-arts.

2. Point Cloud Coding: A Brief Review

The compression of the 3D point cloud data is a hot topic and preliminary investigations have been recently made.

2.1. Volumetric/Tree-Based Point Clouds Coding

For unstructured point clouds, Octree representation is commonly utilized as a point-cloud geometry compression method. To improve the immersive visual experience, De Oliveira Renteet et al. [18] proposed an efficient geometric coding scheme for point clouds, in which an octree-based compression algorithm compression was utilized as a basic layer and used the graph transformation technique as an enhancement layer to encode the residual data. Their reported evaluation results show that the method produced significant improvement, especially at low and medium bit rates. Traditional point-cloud compression algorithms are limited to encoding the position and attribute of discrete point clouds. In [19], Krivokua et al. introduced an alternative technique based on volume function. Like regression analysis, the volume function is a continuous function that can interpolate the values on a finite set of points into a linear combination of continuous basis functions. B-spline wavelet basis is utilized for encoding the volume function, which represents the geometry and attributes of point clouds. Compared to the latest MPEG point-cloud coding standard [20], their algorithm achieves better coding performance in geometry and attributes. To alleviate the computational complexity of the 3D point-cloud model in registration, data abstraction, and visualization, Elseberg et al. [21] proposed an effective point-cloud storage and compression scheme based on the Octree. The coding scheme can be used in file format conversion and 3D model registration.

2.2. Image/Video-Based Point Clouds Coding

For structured point clouds, some studies have focused on employing image/video codecs to compress point cloud data by mapping it into 2D data. Similar to this approach, Tu et al. [22] transformed LiDAR point clouds into a range image sequence and used a simultaneous localization and mapping (SLAM) algorithm to perform inter-prediction. The intra-frame and inter-prediction residual data are encoded by the MPEG-like compression method. Unlike the aforementioned methods, Wang et al. [23] proposed a method to compress RGB-D point clouds. The properties between RGB-D point clouds and LiDAR point clouds are similar, except for the measurement range (i.e., Lidar is around 100 m, while RGB-D camera is around a few meters). They developed a warping-based depth data coding method, in which a point-cloud registration algorithm was utilized to remove redundancy. Experimental results showed that their algorithm achieved a higher compress

ratio with less distortion compared to recent methods. Tu et al. [24] used the conventional image and video-based schemes to compress the 2D arrays by converting the LiDAR data to a range image. Feng et al. [25] proposed a real-time spatio-temporal LiDAR point clouds compression scheme. In this scheme, key frames are identified and encoded by interative plane fitting, and then the temporal streams are encoded by referencing the spatially encoded data. In [26], Tu et al. firstly chose frames as keyframes (I-frame) and obtained the optical flow between the two nearest keyframes. Then, according to the two keyframes and the optical flow, a U-net network was utilized to generate the remaining LiDAR frames (P-frames) between the two keyframes. They removed the temporal redundancy by interpolating point cloud data between two non-adjacent frames. In [27], Tu et al. proposed an RNN-based network to encode LiDAR point clouds. They used a recurrent neural network, and only the input of the first layer was the point cloud data, while the inputs of other layers were the residual data. Their method was to remove the spatial redundancy of a frame of point clouds, not the temporal redundancy. Coding each frame is independent and does not depend on other frames.

2.3. Summary

Currently, there is still no efficient coding solution for the LiDAR point cloud map. Though volumetric/tree-based schemes, such as Google Draco [28] and MPEG TMC 13 [20], can be used to encode the LiDAR point cloud map, their coding performance is far from satisfactory as these methods fail to utilize the spatial structure of point clouds. Converting LiDAR data into a 2D matrix in itself is an efficient method for reducing spatial redundancy. However, most of the existing methods [22,24,29] directly use the image/video-based method to encode LiDAR range image without further exploiting the temporal and spatial redundancies.

3. Overall Codec Architecture

In the paper, the problem of efficient coding dense LiDAR point cloud map is addressed, which can be used for mobile robots or surveying and mapping fields. The low-drift and real-time LOAM algorithm is utilized to construct a LiDAR point cloud map [17]. During the mapping process, the time stamp of the LiDAR scans and quaternion matrices is recorded relative to the global coordinate origin. As the LiDAR point cloud map is constructed by these frames, the dense point cloud map compression can be specified into a point cloud sequence coding task.

Generally, the point cloud sequence compression needs to exploit both temporal and spatial redundancies. The system architecture of our LiDAR point cloud map coding algorithm is illustrated in Figure 1. We divide the frames in the point cloud sequence into intra-frames (I-frames) and inter-frames (B-frames, bi-prediction). An I-frame is compressed independently by removing the spatial redundancies, while a B-frame is compressed by referring to encoded I-frames or B-frames to remove the temporal redundancies. Two I-frames are encoded firstly, followed by B-frames in the middle of two I-frames. B-frames cannot be encoded independently and rely on two encoded I-frames [30].

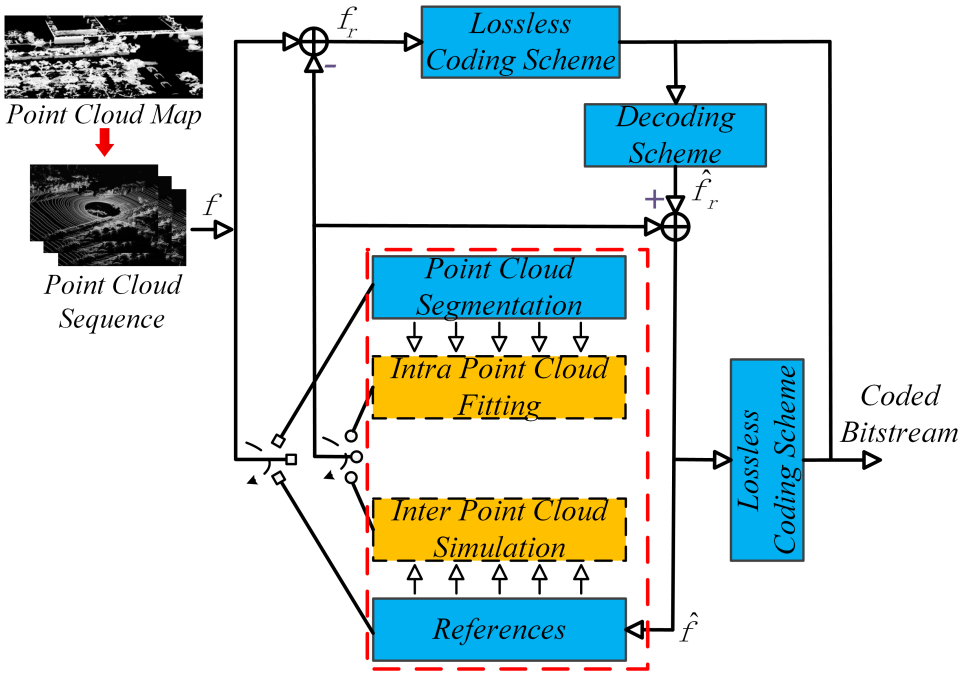

Figure 1. System architecture of our LiDAR point cloud map encoder.

The intra-frames are coded by a segmentation-based prediction technique, while for inter-frames, we develop an interpolation-based coding network to remove the temporal redundancy.

Decoding is the inverse process of encoding. The decoded residual data is added to the prediction data to recover the point clouds. According to the decoded point clouds, quaternion matrices, and translation matrices, the point cloud map can be reconstructed:

$$\begin{bmatrix} x_B \\ y_B \\ z_B \end{bmatrix} = R_{yaw} \times R_{pitch} \times R_{roll} \times \begin{bmatrix} x_I \\ y_I \\ z_I \end{bmatrix} + \begin{bmatrix} C_x \\ C_y \\ C_z \end{bmatrix}, \quad (1)$$

where x_I, y_I, z_I is x, y, z of coordinate system of an intra frame, and x_B, y_B, z_B are corresponding x, y, z, when using the predicted B-frame as the coordinate origin. C_x, C_y, and C_z denotes the translation matrix, and R_{yaw}, R_{pitch}, R_{roll} represents the rotation matrix of yaw, pitch, and roll angle.

4. Intra-Frame Point Cloud Coding Based on Semantic Segmentation

4.1. Overview of Intra-Coding Network

The pipeline of the proposed segmentation-based intra-frame point cloud coding method is illustrated in Figure 2. Firstly, the point cloud conversion from 3D to 2D is performed to obtain the 2D Matrix of point clouds. Then, the RangeNet++ [31] network is utilized to realize point cloud segmentation. The residual data, contour map, and the quadric surface parameters are encoded with lossless coding schemes and packaged as the the intra-bitstream.

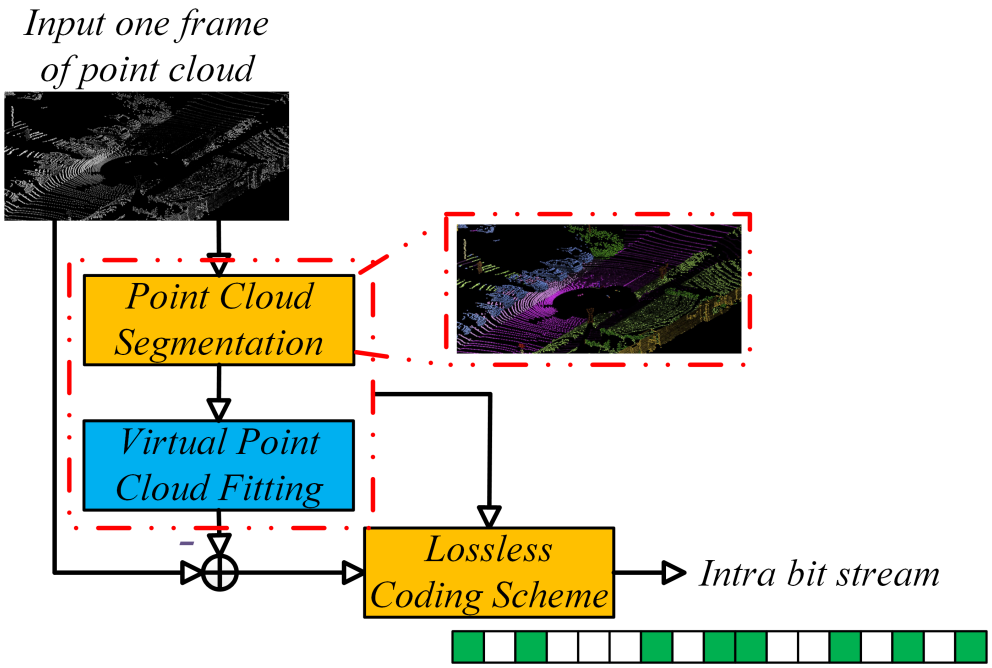

Figure 2. Intra-prediction technique based on segmentation.

4.2. LiDAR Point Cloud Segmentation

The LiDAR data from the KITTI dataset is utilized to verify our method, which uses 64 channels, covering 26.9° vertical field of view and a 360° horizontal field of view [32]. Considering that LiDAR sensors acquire data in an orderly way, the point clouds can be converted from $\mathbb{R}^3$ to $\mathbb{R}^2$. Let $(I_i = (x_i, y_i, z_i))_{i=1...N}$ be the coordinates of a frame of point cloud captured by Velodyne sensors. Considering that the point clouds is ordered, it can be transformed into a 2D matrix $X(u, v)_{u=1...M, v=1...N}$.

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{1}{2}(1 - arctan(y, x)\pi^{-1}) \cdot w \\ (1 - (arcsin(zr^{-1} + f_{up}))f^{-1}) \cdot h \end{pmatrix}, \tag{2}$$

where (x, y, z) represent the coordinates of point P, (u, v) are 2D matrix coordinates, (h, w) are the height and width of the desired 2D matrix representation, $r = \sqrt{x^2 + y^2 + z^2}$ represents the distance of point $P = (x, y, z)$ from the origin, and $f = f_{up} + f_{down}$ denotes the vertical field of view of the LiDAR sensor [33].

The RangeNet++ takes the fused 2D information as input and outputs the segmentation results. Three 2D convolutional blocks are adopted as a 2D feature extractor. The output of the final layer is the point cloud segmentation results. The segmentation results will pave the way for the subsequent surface fitting-based intra-prediction technique.

4.3. Segmentation-Based Intra-Prediction Technique

Nearly one-third of the point cloud data are ground points. After segmenting the point cloud, we can segment the ground and other objects. Thus, the quadric surface fitting-based technique is utilized to fit the point clouds. Considering the complexity and compression efficiency, we fit the segmented region with the plane for the ground points and sphere for other object.

A plane is defined by $J = (n, d)$, where n is the normal vector, with $||n|| = 1$, and d is the vertical distance from the origin to the plane. The distance from a point p_i to the plane is defined as:

$$D_{plane}(J, p_i) = ||n^T p - d|| \tag{3}$$

Then we can construct the equation ε_{plane} as a function of the distance of the minimum value of the sum.

$$\varepsilon_{plane}(J,S) = \sum_{i=1}^{N} ||n^T S_{i,:} - d|| \tag{4}$$

A sphere is represented by $J = (c,r)$, where $r \in \mathcal{R}$ denotes the radius, and $c \in \mathcal{R}^3$ represents the center. The distance from point p_i to the sphere is defined as:

$$D_{sphere}(J,p_i) = ||c - p_i|| - r \tag{5}$$

Then we can construct the fitting equation ε_{sphere} as a function of the distance of the minimum value of the sum.

$$\varepsilon_{sphere}(J,S) = \sum_{i=1}^{N} ||(||c - p_i||) - r|| \tag{6}$$

4.4. Residual Data Coding

According to the parameters of the LiDAR sensor, we use the fitting plane to calculate the coordinates of the virtual points. The residual data $R_{intra}(x,y)$ is the difference between the original range image $X_{intra}(x,y)$ and the predicted range image $P_{intra}(x,y)$. As the pixel values of the residual data are nearly zero, the entropy of the $R_{intra}(x,y)$ is smaller copared with $X_{intra}(x,y)$. Thus, the residual data $R_{intra}(x,y)$ can be encoded with fewer bits.

$$R_{intra}(x,y) = X_{intra}(x,y) - P_{intra}(x,y). \tag{7}$$

5. Inter-Frame Point Cloud Coding Based on Inserting Network

5.1. Overall Inter-Prediction Network

To remove temporal redundancy in the LiDAR point clouds [34], an inter-frame point cloud inserting network is designed, as illustrated in Figure 3. The interpolation module utilizes the encoded points clouds $X = \{X_{t0}, X_{t0+2k}\}$, and generates the internal point cloud frame P_{t0+k}. We calculate the difference between the predicted result P_{t0+k} and the real point cloud X_{t0+k} as residual data $R_{inter}(x,y)$, which will be encoded as the inter-bitstream.

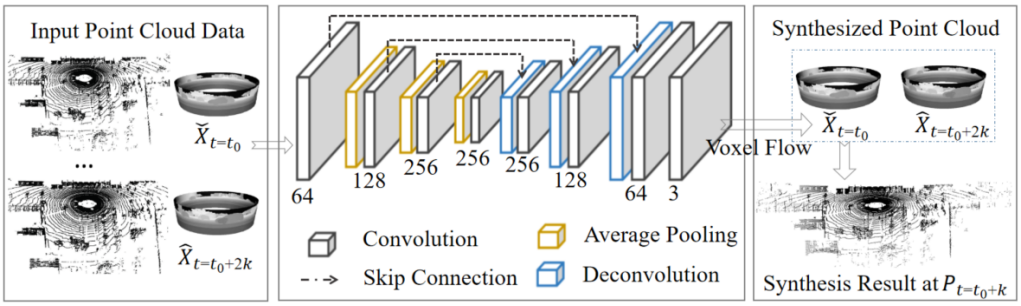

Figure 3. Inter-frame point cloud inserting network.

5.2. Point Cloud Interpolation Module

Figure 3 illustrates the diagram of the LiDAR point cloud interpolation module. The encoder and decoder parts predict the 3D voxel stream, which will be used to generate the required intermediate frames. The network generate the predicted frame $P_{t=t_0+k}$ according to the input point clouds $\check{X}_{t=t_0}$ and $\hat{X}_{t=t_0+2k}$, where $\check{X}_{t=t_0}$, $\hat{X}_{t=t_0+2k}$ are already encoded frames. The predicted frame can be the middle frame by interpolating or the next frame by extrapolating the input point clouds. We focus on interpolating the intermediate frame according to two decoded frames. The network is represented by $H(X_{rec},\Theta)$, where the output F is the 3D voxel flow of the input X_{rec}, and Θ is the network parameters.

$$F = (\Delta x, \Delta y, \omega) = H(X_{rec};\Theta), \tag{8}$$

where F is the optical flow of two adjacent frames. The opposite direction of the optical flow is used to identify the corresponding position in the previous frame. The coordinates of the corresponding positions in the preceding and following frames is defined as $L_{former} = (x - \Delta x, y - \Delta y)$ and $L_{later} = (x + \Delta x, y + \Delta y)$. We use tri-linear interpolation from the eight corner points of the voxel to calculate the output value $P(x, y)$ by four points: $\check{X}_{00}(\check{x}_{ceil}, \check{y}_{ceil})$, $\check{X}_{01}(\check{x}_{ceil}, \check{y}_{floor})$, $\check{X}_{10}(\check{x}_{floor}, \check{y}_{ceil})$, and $\check{X}_{11}(\check{x}_{floor}, \check{y}_{floor})$, from the former frame, and the other four points: $\hat{X}_{00}(\hat{x}_{ceil}, \hat{y}_{ceil})$, $\hat{X}_{01}(\hat{x}_{ceil}, \hat{y}_{floor})$, $\hat{X}_{10}(\hat{x}_{floor}, \hat{y}_{ceil})$ and $\hat{X}_{11}(\hat{x}_{floor}, \hat{y}_{floor})$, from the later frame.

The time component of the voxel stream F can be considered as a linear blending weight between two adjacent frames. We employ this voxel stream to sample the input two frames and use the volume sampling function $T_{x,y,\omega}$ to generate the final predicted frame P.

$$\begin{aligned} P_{inter}(x, y) &= T_{x,y,\omega}(X_{rec}, H(X_{rec}; \Theta)) \\ &= \omega \cdot P_{former}(x, y) + (1 - \omega) \cdot P_{later}(x, y), \end{aligned} \tag{9}$$

where $P_{former}(x, y)$ and $P_{later}(x, y)$ are computed by

$$\begin{aligned} P_{former}(x, y) &= \begin{bmatrix} 1 - x & x \end{bmatrix} \times \begin{bmatrix} \check{X}_{00} & \check{X}_{01} \\ \check{X}_{10} & \check{X}_{11} \end{bmatrix} \times \begin{bmatrix} 1 - y \\ y \end{bmatrix}, \\ P_{later}(x, y) &= \begin{bmatrix} 1 - x & x \end{bmatrix} \times \begin{bmatrix} \hat{X}_{00} & \hat{X}_{01} \\ \hat{X}_{10} & \hat{X}_{11} \end{bmatrix} \times \begin{bmatrix} 1 - y \\ y \end{bmatrix}. \end{aligned} \tag{10}$$

The interpolation network adopts a fully convolutional structure, with four convolutional layers and four deconvolutional layers. To better maintain the spatial features in the low dimension layer, some jump connections between the corresponding convolutional layer and deconvolutional layer are added.

5.3. Inter Loss Function Design

The prediction module is represented by $H(X_{(t0,t0+2k)}, \Theta)$, where the output P_{t0+k} is the predicted point cloud at $t = T + 1$ and Θ is the network parameters.

$$P_{t0+k} = H(X_{(t0,t0+2k)}, \Theta), \tag{11}$$

The predited point cloud data are converted to range image, and the difference between the original range image $X_{t0+k}(x, y)$ and the predicted range image $P_{t0+k}(x, y)$ is calculated as the residual data $R_{inter}(x, y)$, which will be encoded losslessly. Explicity, the training loss is defined as follows:

$$\begin{aligned} R_{inter}(x, y) &= X_{t0+k}(x, y) - P_{t0+k}(x, y) \\ &= X_{t0+k}(x, y) - H(X_{(t0,t0+2k)}, \Theta) \end{aligned} \tag{12}$$

$$L_{loss} = ||R_{inter}||_2 \tag{13}$$

5.4. Visualization Results

For the inter-coding network, if the point-cloud interpolation module synthesizes more accurate point clouds, smaller residual data and higher compression performance can be achieved. To deeply evaluate, four scenes point cloud interpolation results are obtained, as illustrated in Figure 4. We can see that the predicted point cloud and the original point cloud are almost the same. The effectiveness of the inter insertion module is confirmed.

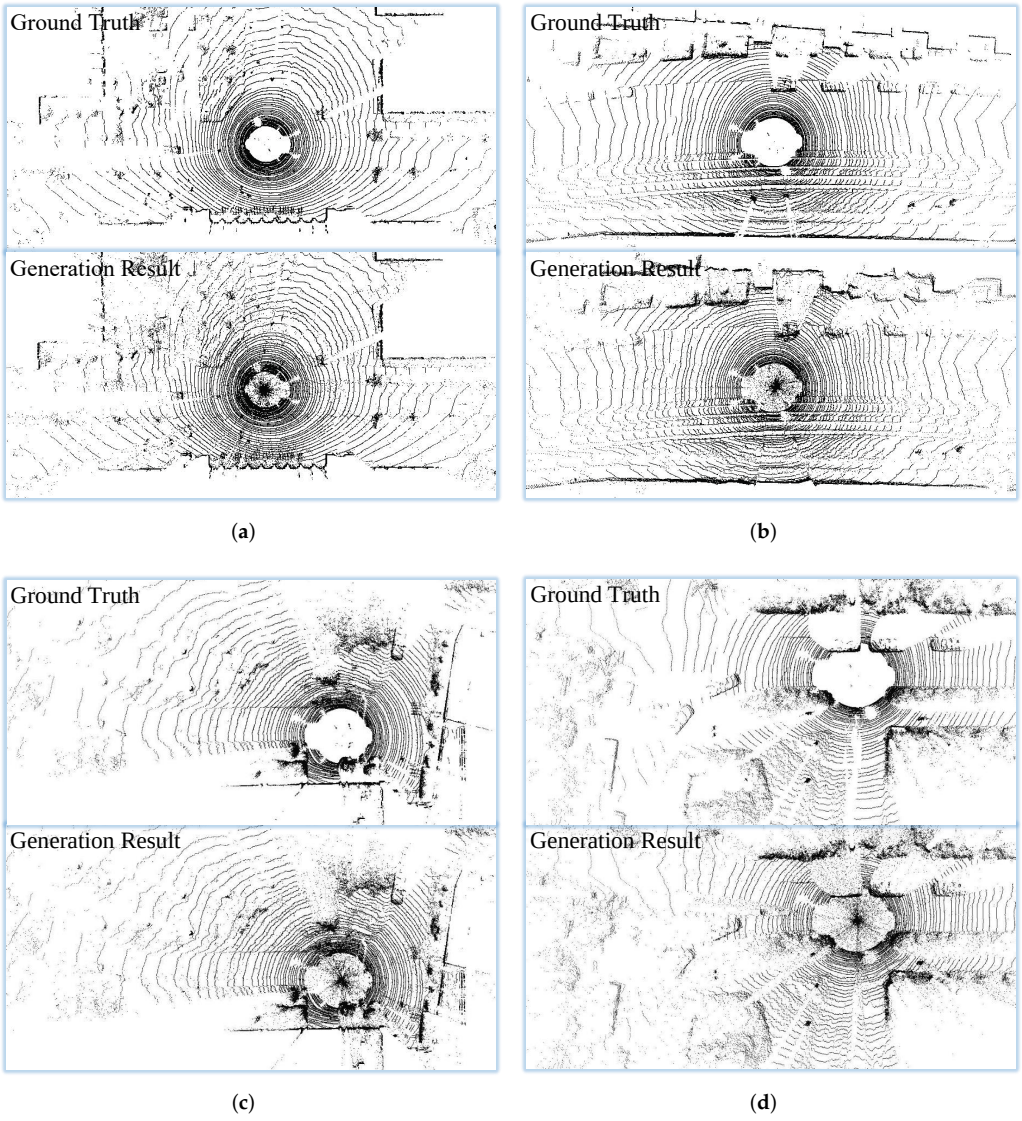

Figure 4. Qualitative results: (**a**) campus; (**b**) city; (**c**) road; (**d**) residential (Best viewed by zooming in).

6. Experimental Results

The proposed point cloud compression scheme is implemented in python using point cloud libraries (PCL) [35] on a PC with a TITAN RTX GPU. The KITTI dataset [36], including city, residential, campus, and road scenes, is used to evaluate our algorithm.

6.1. Evaluation Metric

To evaluate the overall performance, the calculation of the compression ratio (CR) and relative distance (D_d) is considered. The CR is obtained by calculating the ratio between the point cloud size after compression and the original size.

$$CR = \frac{Compressed_{size}}{Original_{size}} \times 100\%, \tag{14}$$

D_d represents the distance between the ground truth LiDAR data P_{input} and the reconstructed one P_{decode}. D_d is defined as follows:

$$D_d(P_{input}, P_{decode}) = \frac{\overline{D}_d(P_{input}, P_{decode})}{2} + \frac{\overline{D}_d(P_{decode}, P_{input})}{2}$$
$$\overline{D}_d(P_i, P_j) = \frac{1}{|P_i|} \sum_{x_i \in P_i} \min_{x_j \in P_j} d(x_i - x_j) \quad (15)$$

The measure is sensitive to false positives (rebuilding points in unoccupied areas) and false negatives (excluding occupied areas).

6.2. Coding Performance for a Single Frame

To find the most efficient coding method for the residual data, several lossless coding schemes are used to encode the residual data, including *Zstandard*, *LZ5*, *Deflate*, *Lizard*, *LZ4*, and *PPMd*. To verify the compression efficiency in removing the time and space redundancy, the 2D matrices are also directly encoded with the lossless coding algorithms without any preprocessing. Figure 5 shows the experimental results.

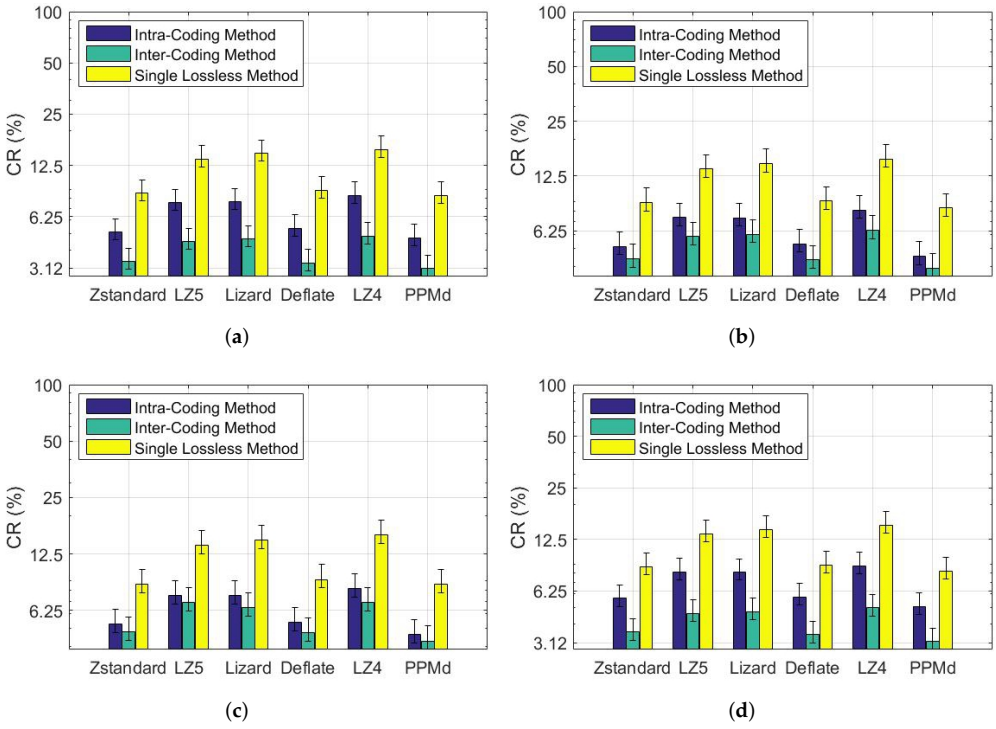

Figure 5. CR of different lossless coding schemes: (**a**) campus; (**b**) city; (**c**) road; (**d**) residential.

For intra-coding, the smallest CR value is 4.64%, achieved by using the *PPMd* scheme for the *city* scene due to its simple structure. The point cloud of the residential scene, however, is complex and the CR is much higher. The smaller the CR, the better the coding performance.

For inter-coding, it can be observed that using *the PPMd* scheme achieves the best compression performance for the *campus* scene, with a CR of 3.09%. The worst CR is 6.87% for the *residential* scene using the *LZ4* scheme. However, the CR of the proposed inter-coding network is still smaller than that of directly coding point cloud data with lossless coding schemes.

6.3. Comparsion with Octree and Draco

According to the time stamp and quaternion matrix of each scan, the scans are merged into a LiDAR panoramic map. Taking this point cloud map as a whole, Table 1 describes the CR results of the proposed coding method compared to *Octree* [35] and Google Draco [28]. The quantization accuracy (QA) of our method is set to 2 mm, 5 mm, and 1 cm, while the distance resolution (DR) of *Octree* is set to 1 mm^3, 5 mm^3, and 1 cm^3. The quantization bits (QB) of Draco are set to 17, 15, and 14, which correspond to 1 mm accuracy, 5 mm accuracy, and 1 cm accuracy, respectively. Besides that, the compression level (CL) = 10 is set to achieve the highest compression rate. In our experiments, *PPMd* is selected to encode residual data. From Table 1, it can be observed that compared with *Octree*, the proposed algorithm achieves a smaller CR value.

Table 1. Comparison ratio results with the *Draco* and *Octree* methods.

	Inter-Inserting Method			***Octree* [35]**			***Draco* [28]**		
Scene	**QA**			**DR**			**QB**		
	2 mm	**5 mm**	**1 cm**	**1 mm^3**	**5 mm^3**	**1 cm^3**	**17 (bits)**	**15 (bits)**	**14 (bits)**
Campus	3.09	2.49	2.01	21.27	8.05	5.75	11.87	7.75	5.47
City	3.90	3.38	2.83	23.98	10.76	8.40	12.52	8.38	6.49
Road	3.16	0.26	2.12	23.56	10.35	7.99	12.31	8.35	6.59
Residential	4.29	3.68	3.16	22.94	9.72	7.37	12.66	8.59	6.33
Average	3.61	3.03	2.53	20.23	9.72	7.29	12.34	8.27	6.22

6.4. Rate-Distortion Curves

Four state-of-the-art baselines are chosen for comparison, including Google Draco [28], MPEG TMC13 [20], and Tu's method [26]. The results are evaluated in terms of the relationship between the Distance and the bit per point (bpp) for point cloud data for four scenes, as illustrated in Figure 6. The proposed LiDAR point clouds coding method has shown outstanding advantages in terms of bbp and D_d among these methods [37–39]. Google Draco and MPEG TMC13 are not designed specifically for multi-line LiDAR data [40,41]. Compared to these methods, our method can generate the point cloud more accurately, which contributes to largely removing the redundancy and obtaining a better D_d-bpp result.

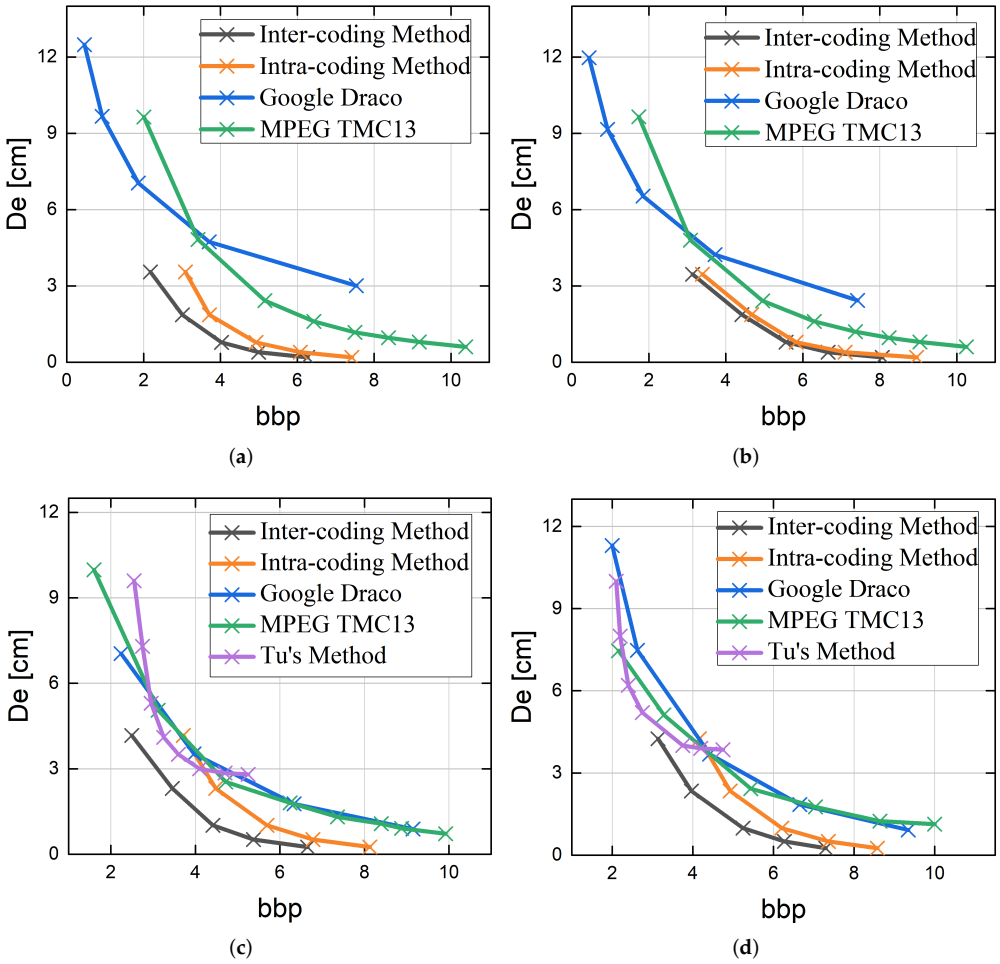

Figure 6. D_d-*bpp* curves of our method in compared to Google Draco [28], MPEG TMC13 [20] and Tu's method [26]: (**a**) *campus*, (**b**) *city*, (**c**) *road*, and (**d**) *residential*.

6.5. Computational Complexity

The proposed intra-coding consists of three steps, namely segmentation, intra-prediction, and residual data coding, while the inter-coding method consists of inserting frame and residual data coding. By calculating the average coding time, 100 frames are selected. Experimental results show that the total intra-coding time with the lossless method (PPMd) is 0.21 s and the total inter-coding time is 0.15 s.

7. Discussion

With a 5–15 HZ user-selectable frame rate, the LiDAR sensor of HDL-64E S2 captures over 1.3 million points per second. The drawback of the proposed LiDAR point cloud map coding scheme is that the intra-coding and inter-coding can not meet the real-time requirement [42–44]. However, it can be used for off-line LiDAR point cloud map coding to reduce its storage space and transmission bandwidth, which can be used in mobile robots or surveying and mapping fields. The follow-up research focuses on implementing the coding scheme to the FPGA platform to accelerate the algorithm and achieve real-time performance.

8. Conclusions

Ranging sensors, such as LiDAR, are considered to be very robust under all light conditions or foggy weather, which have been widely used in the field of autonomous driving tasks, for instance, navigation, obstacle avoidance, target tracking, and recognition, etc. However, the enormous volume of LiDAR point clouds brings great challenges to

data storage and transmission. To address this issue, this paper focuses on LiDAR point cloud map coding. Firstly, an intra-coding technique is designed based on point cloud segmentation and geometric reconstruction, which can effectively remove the spatial redundancy of the LiDAR point cloud. Secondly, we designed a point cloud insertion network to remove the time redundancy of point clouds by inserting frames into the intermediate moment according to the encoded point clouds. Experiments demonstrate that the proposed method obtains a higher comparison performance compared with several representative point cloud methods.

Author Contributions: Conceptualization, Q.W. and L.J.; methodology, X.S.; software, Z.D. and S.Y.; validation, Q.W., L.J. and J.Z.; formal analysis, Q.W., J.Z. and S.Y.; investigation, X.S. and Q.W.; resources, Q.W. and X.S.; data curation, X.S.; writing—original draft preparation, Q.W. and X.S.; writing—review and editing, Q.W. and S.Y.; visualization, Z.D. and L.J.; supervision, S.Y.; project administration, X.S.; Funding acquisition, S.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This study is supported by the National Natural Science Foundation of China (Grant No. 62001262) and the National Natural Science Foundation of China (Grant No. 62001263) and the Nature Science Foundation of Shandong Province (Grant No. ZR2020QF008).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wang, C.; Ji, M.; Wang, J.; Wen, W.; Li, T.; Sun, Y. An improved DBSCAN method for LiDAR data segmentation with automatic Eps estimation. *Sensors* **2019**, *19*, 172. [CrossRef] [PubMed]
2. McGlade, J.; Wallace, L.; Reinke, K.; Jones, S. The potential of low-cost 3D imaging technologies for forestry applications: Setting a research agenda for low-cost remote sensing inventory tasks. *Forests* **2022**, *13*, 204. [CrossRef]
3. Guimarães, N.; Pádua, L.; Marques, P.; Silva, N.; Peres, E.; Sousa, J.J. Forestry remote sensing from unmanned aerial vehicles: A review focusing on the data, processing and potentialities. *Remote Sens.* **2020**, *12*, 1046. [CrossRef]
4. Sadeghifar, T.; Lama, G.; Sihag, P.; Bayram, A.; Kisi, O. Wave height predictions in complex sea flows through soft-computing models: Case study of Persian Gulf. *Ocean Eng.* **2022**, *245*, 110467. [CrossRef]
5. Gale, M.G.; Cary, G.J.; Van Dijk, A.I.; Yebra, M. Forest fire fuel through the lens of remote sensing: Review of approaches, challenges and future directions in the remote sensing of biotic determinants of fire behaviour. *Remote Sens. Environ.* **2021**, *255*, 112282. [CrossRef]
6. Jalonen, J.; Järvelä, J.; Virtanen, J.P.; Vaaja, M.; Kurkela, M.; Hyyppä, H. Determining characteristic vegetation areas by terrestrial laser scanning for floodplain flow modeling. *Water* **2015**, *7*, 420–437. [CrossRef]
7. Lama, G.F.C.; Sadeghifar, T.; Azad, M.T.; Sihag, P.; Kisi, O. On the indirect estimation of wind wave heights over the southern coasts of Caspian Sea: A comparative analysis. *Water* **2022**, *14*, 843. [CrossRef]
8. Godone, D.; Allasia, P.; Borrelli, L.; Gullà, G. UAV and structure from motion approach to monitor the maierato landslide evolution. *Remote Sens.* **2020**, *12*, 1039. [CrossRef]
9. He, J.; Barton, I. Hyperspectral remote sensing for detecting geotechnical problems at Ray mine. *Eng. Geol.* **2021**, *292*, 106261. [CrossRef]
10. Rao, Y.; Zhang, M.; Cheng, Z.; Xue, J.; Pu, J.; Wang, Z. Semantic Point Cloud Segmentation Using Fast Deep Neural Network and DCRF. *Sensors* **2021**, *21*, 2731. [CrossRef]
11. Xue, G.; Wei, J.; Li, R.; Cheng, J. LeGO-LOAM-SC: An Improved Simultaneous Localization and Mapping Method Fusing LeGO-LOAM and Scan Context for Underground Coalmine. *Sensors* **2022**, *22*, 520. [CrossRef] [PubMed]
12. Xiong, L.; Fu, Z.; Zeng, D.; Leng, B. An optimized trajectory planner and motion controller framework for autonomous driving in unstructured environments. *Sensors* **2021**, *21*, 4409. [CrossRef] [PubMed]
13. Vanhellemont, Q.; Brewin, R.J.; Bresnahan, P.J.; Cyronak, T. Validation of Landsat 8 high resolution Sea Surface Temperature using surfers. *Estuar. Coast. Shelf Sci.* **2021**, *265*. 107650 [CrossRef]
14. Walton, C.C. A review of differential absorption algorithms utilized at NOAA for measuring sea surface temperature with satellite radiometers. *Remote Sens. Environ.* **2016**, *187*, 434–446. [CrossRef]

15. Lama, G.F.C.; Rillo Migliorini Giovannini, M.; Errico, A.; Mirzaei, S.; Padulano, R.; Chirico, G.B.; Preti, F. Hydraulic efficiency of green-blue flood control scenarios for vegetated rivers: 1D and 2D unsteady simulations. *Water* **2021**, *13*, 2620. [CrossRef]
16. Pomerleau, F.; Liu, M.; Colas, F.; Siegwart, R. Challenging data sets for point cloud registration algorithms. *Int. J. Robot. Res.* **2012**, *31*, 1705–1711. [CrossRef]
17. Zhang, J.; Singh, S. Low-drift and real-time lidar odometry and mapping. *Auton. Robot.* **2017**, *41*, 401–416. [CrossRef]
18. 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]
19. Krivokuća, M.; Chou, P.A.; Koroteev, M. A volumetric approach to point cloud compression—Part ii: Geometry compression. *IEEE Trans. Image Process.* **2019**, *29*, 2217–2229. [CrossRef]
20. Guede, C.; Andrivon, P.; Marvie, J.E.; Ricard, J.; Redmann, B.; Chevet, J.C. V-PCC: Performance evaluation of the first MPEG Point Cloud Codec. In Proceedings of the SMPTE 2020 Annual Technical Conference and Exhibition, Virtual, 10–12 November 2020; pp. 1–27.
21. Elseberg, J.; Borrmann, D.; Nüchter, A. One billion points in the cloud–an octree for efficient processing of 3D laser scans. *ISPRS J. Photogramm. Remote Sens.* **2013**, *76*, 76–88. [CrossRef]
22. Tu, C.; Takeuchi, E.; Carballo, A.; Miyajima, C.; Takeda, K. Motion analysis and performance improved method for 3D LiDAR sensor data compression. *IEEE Trans. Intell. Transp. Syst.* **2019**, *22*, 243–256. [CrossRef]
23. Wang, X.; Şekercioğlu, Y.A.; Drummond, T.; Natalizio, E.; Fantoni, I.; Frémont, V. Fast depth video compression for mobile RGB-D sensors. *IEEE Trans. Circuits Syst. Video Technol.* **2015**, *26*, 673–686. [CrossRef]
24. Tu, C.; Takeuchi, E.; Miyajima, C.; Takeda, K. Compressing continuous point cloud data using image compression methods. In Proceedings of the 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), Janeiro, Brazil, 1–4 November 2016; pp. 1712–1719.
25. Feng, Y.; Liu, S.; Zhu, Y. Real-time spatio-temporal lidar point cloud compression. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 25–29 October 2020; pp. 10766–10773.
26. Tu, C.; Takeuchi, E.; Carballo, A.; Takeda, K. Real-time streaming point cloud compression for 3d lidar sensor using u-net. *IEEE Access* **2019**, *7*, 113616–113625. [CrossRef]
27. 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 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 3274–3280.
28. Google. Draco: 3D Data Compression. 2018. Available online: https://github.com/google/draco (accessed on 6 May 2022).
29. Houshiar, H.; Nüchter, A. 3D point cloud compression using conventional image compression for efficient data transmission. In Proceedings of the 2015 XXV International Conference on Information, Communication and Automation Technologies (ICAT), Washington, DC, USA, 29–31 October 2015; pp. 1–8.
30. Liu, Z.; Yeh, R.A.; Tang, X.; Liu, Y.; Agarwala, A. Video frame synthesis using deep voxel flow. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 4463–4471.
31. Milioto, A.; Vizzo, I.; Behley, J.; Stachniss, C. Rangenet++: Fast and accurate lidar semantic segmentation. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 3–8 November 2019; pp. 4213–4220.
32. 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.* **2020**, 23, 2190–2201. [CrossRef]
33. Langer, F.; Milioto, A.; Haag, A.; Behley, J.; Stachniss, C. Domain Transfer for Semantic Segmentation of LiDAR Data using Deep Neural Networks. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 25–29 October 2020.
34. Sun, X.; Wang, S.; Wang, M.; Cheng, S.S.; Liu, M. An advanced LiDAR point cloud sequence coding scheme for autonomous driving. In Proceedings of the 28th ACM International Conference on Multimedia, Seattle, WA, USA, 12–16 October 2020; pp. 2793–2801.
35. Rusu, R.B.; Cousins, S. 3D is here: Point cloud library (PCL). In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 1–4.
36. Geiger, A.; Lenz, P.; Stiller, C.; Urtasun, R. Vision meets robotics: The KITTI dataset. *Int. J. Robot. Res.* **2013**, *32*, 1231–1237. [CrossRef]
37. Yang, B.; Li, J. A hierarchical approach for refining point cloud quality of a low cost UAV LiDAR system in the urban environment-ScienceDirect. *ISPRS J. Photogramm. Remote Sens.* **2022**, *183*, 403–421. [CrossRef]
38. He, P.; Emami, P.; Ranka, S.; Rangarajan, A. Learning Scene Dynamics from Point Cloud Sequences. *Int. J. Comput. Vis.* **2022**, *130*, 669–695. [CrossRef]
39. Zhou, B.; He, Y.; Huang, W.; Yu, X.; Fang, F.; Li, X. Place recognition and navigation of outdoor mobile robots based on random Forest learning with a 3D LiDAR. *J. Intell. Robot. Syst.* **2022**, *104*, 267–279. [CrossRef]
40. Shi, Y.; Yang, C. Point cloud inpainting with normal-based feature matching. *Multimed. Syst.* **2022**, *28*, 521–527. [CrossRef]
41. Tian, J.; Zhang, T. Secure and effective assured deletion scheme with orderly overwriting for cloud data. *J. Supercomput.* **2022**, *78*, 9326–9354. [CrossRef]

42. Luan, S.; Chen, C.; Zhang, B.; Han, J.; Liu, J. Gabor Convolutional Networks. *IEEE Trans. Image Process.* **2017**, *2018*, 4357–4366.
43. Zhang, B.; Perina, A.; Li, Z.; Murino, V.; Liu, J.; Ji, R. Bounding multiple gaussians uncertainty with application to object tracking. *Int. J. Comput. Vis.* **2016**, *118*, 364–379. [CrossRef]
44. Zhang, B.; Yang, Y.; Chen, C.; Yang, L.; Han, J.; Shao, L. Action recognition using 3D histograms of texture and a multi-class boosting classifier. *IEEE Trans. Image Process.* **2017**, *26*, 4648–4660. [CrossRef] [PubMed]