algorithms

MDPI

Article

TransPCGC: Point Cloud Geometry Compression Based on Transformers

Shiyu Lu, Huamin Yang * and Cheng Han

School of Computer Science and Technology, Changchun University of Science and Technology, Changchun 130022, China; 2020200128@mails.cust.edu.cn (S.L.); hancheng@cust.edu.cn (C.H.)
* Correspondence: yanghuamin@cust.edu.cn

Abstract: Due to the often substantial size of the real-world point cloud data, efficient transmission and storage have become critical concerns. Point cloud compression plays a decisive role in addressing these challenges. Recognizing the importance of capturing global information within point cloud data for effective compression, many existing point cloud compression methods overlook this crucial aspect. To tackle this oversight, we propose an innovative end-to-end point cloud compression method designed to extract both global and local information. Our method includes a novel Transformer module to extract rich features from the point cloud. Utilization of a pooling operation that requires no learnable parameters as a token mixer for computing long-distance dependencies ensures global feature extraction while significantly reducing both computations and parameters. Furthermore, we employ convolutional layers for feature extraction. These layers not only preserve the spatial structure of the point cloud, but also offer the advantage of parameter independence from the input point cloud size, resulting in a substantial reduction in parameters. Our experimental results demonstrate the effectiveness of the proposed TransPCGC network. It achieves average Bjontegaard Delta Rate (BD-Rate) gains of 85.79% and 80.24% compared to Geometry-based Point Cloud Compression (G-PCC). Additionally, in comparison to the Learned-PCGC network, our approach attains an average BD-Rate gain of 18.26% and 13.83%. Moreover, it is accompanied by a 16% reduction in encoding and decoding time, along with a 50% reduction in model size.

Keywords: point cloud geometry compression; transformers; convolution

Citation: Lu, S.; Yang, H.; Han, C. TransPCGC: Point Cloud Geometry Compression Based on Transformers. *Algorithms* **2023**, *16*, 484. https://doi.org/10.3390/a16100484

Academic Editor: Binhai Zhu

Received: 10 September 2023
Revised: 6 October 2023
Accepted: 16 October 2023
Published: 19 October 2023

1. Introduction

With the rapid advancement of real-time sensors such as depth cameras and LiDAR, point clouds have found widespread applications in Simultaneous Location And Mapping (SLAM) [1], 3D reconstruction [2], object detection [3], and more. Spatial data collection has been considerably improved with the invention of LiDAR and other laser scanning technologies [4]. However, the increased volume of point clouds leads to higher transmission costs, making it imperative to research point cloud compression algorithms. Point cloud geometry compression methods can be broadly classified into two categories: lossy geometry compression and lossless geometry compression. Lossy geometric compression algorithms have been extensively studied due to their high compression ratio properties [5–8]. Lossless geometry compression aims to strike a balance between compressed data size and data quality [9–13]. When more bits are used, it is possible to obtain a value closer to the input. This concept is formalized in the basic rate-distortion tradeoff, where "rate" stands for bit rate and "distortion" is formalized as the difference between the original and disordered point clouds.

The sparsity and disorder of point clouds pose significant challenges to data processing and compression. Consequently, point clouds are converted into structured representations, such as tree structures, heightmaps, or voxel representations, which are then compressed using their inherent spatial correlation. With the emergence of deep learning-based methods,

several studies have explored neural network-based point cloud compression. Previous research [5–15] focused on voxel-based approaches, while [16–18] utilized tree structures, and [19] employed a heightmap representation. Deep learning-based methods achieve superior compression performance compared to traditional algorithms by learning more memory-efficient encoding strategies from training data. However, there is still space for improvement in these methods.

This paper presents a point cloud geometry compression based on the Transformer method (TransPCGC). The main contributions of this scheme are as follows:

- We design a novel Transformer-based network for compressing geometric information in point clouds. This network comprises preprocessing modules for voxelization and partitioning of point clouds, compression networks to optimize the BD-Rate, and post-processing modules for point cloud reconstruction. In contrast to pure convolutional neural networks, our designed network achieves remarkable compression gains.
- The effectiveness of point cloud compression is hindered due to convolutions designed with fixed receptive fields and fixed weights, which cannot aggregate enough information from sparse and disordered point clouds. There is a need for a solution that can better exploit correlations among global and local information point clouds. As the Transformer architecture designed for processing images is not well suited for point clouds, we develop a Transformer capable of handling point clouds. Utilizing this Transformer, we extract the richness of both global and local features from the point cloud.
- Extensive experiments validate the compression performance of the proposed TransPCGC on multiple point cloud datasets, while also demonstrating its low spatial and temporal complexity.

The remainder of this paper is as follows: Section 2 provides a summary overview of the related work; Section 3 describes our proposed compression and decompression network; Section 4 offers the experimental details and presents the experimental results; Section 5 briefly concludes the paper.

2. Related Work

2.1. Non-Deep Learning Methods

Over the past few years, a series of Point Cloud Compression (PCC) standards have been developed under the MPEG consortium. Prominent examples include the geometry-based PCC (G-PCC) v14 [20] and V-PCC [21]. Meshes, voxels, and point clouds serve as widely used 3D representation forms. Mesh models consist of vertices and faces formed by the connections/topology between vertices. Mesh compression methods [22] save topology memory through intermediate representations, where vertices can also be compressed using topological information, resulting in improved subjective quality with fewer encoding bits. For geometry compression based on octrees, the octree structure is commonly utilized to provide better context for entropy coding. Point cloud encoding is executed through manually defined rules and data structures. The octree geometry codec in PCL [23] and MPEG standard G-PCC [24] are two popular octree-based methods. Google's Draco [25] employs a kd-tree to partition space and encodes points based on the occupancy of partitioned space. Some algorithms do not directly compress the 3D geometric structure of point clouds; they project the point clouds onto 2D images and apply video compression algorithms, as seen in the V-PCC method in the MPEG standard [20]. Dumic et al. [26] presented a dynamic point cloud compression based on different projection types and bit depth combined with the surface reconstruction algorithm and video compression for obtained geometry and texture maps. Yu et al. [27] proposed an improved compression means for dynamic point cloud based on curvature estimation and hierarchical strategy. Certain works [28] also explored the compression of dynamic point cloud sequences by introducing motion estimation between consecutive frames. These non-learning-based methods encode coordinates via predefined rules and generally exhibit robustness across different types of point clouds, but their encoding performance may be limited.

2.2. Deep Learning Methods

With the development of deep learning architectures, networks have recently been created for the autoencoder-based reconstruction [29,30], completion [31], or production of point clouds [32]. Learning-based geometric compression techniques have been developed to enhance compression performance by acquiring efficient representations from point clouds, leveraging point cloud autoencoder techniques. Some of these methods involve converting raw point clouds into voxels and subsequently employing a 3D CNN to encode the geometry into a binary form. Quach et al. [5] introduced a geometric information compression algorithm based on convolutional operations and uniform quantization. This approach utilizes a tradeoff parameter for joint optimization of rate and distortion, transforming the decoding process into a binary classification problem of point cloud occupancy. Subsequently, an improved version was proposed, incorporating entropy coding with hyperprior modeling, optimal threshold decoding, and sequential model training [6]. Wang et al. [7] presented an end-to-end learning-based algorithm for efficient compression of point cloud geometric information using a stacked convolutional neural network-based variational autoencoder. They also leveraged the sparsity of point clouds to perform progressive resampling for hierarchical point cloud reconstruction [8] and further proposed voxel compression using inter-scale and intra-scale correlations [9]. Researchers like André [14] and others [10] enhanced compression performance by adding modules to the multi-scale point cloud geometry compression network. Yu et al. [33] proposed a multi-layer residual module designed on sparse convolution-based autoencoders which progressively down-samples the input point clouds and hierarchically reconstructs them. Zhuang et al. [34] introduced a rate expansion method based on contrastive learning to expand the bit rate range of the model.

Nguyen et al. [10] introduced a lossless compression method for point cloud geometry information, leveraging neural networks and context-adaptive arithmetic coding. To address the slow inference caused by predicting occupancy probability in voxel order in prior designs, a multi-scale architecture was proposed, optimizing voxel occupancy modeling from coarse to fine order [11]. They subsequently suggested using neural network estimation of voxel occupancy probability distribution [12]. Wang and Nguyen later delved into attribute compression of point clouds [13,15]. You et al. [35] proposed a patch-based point cloud compression algorithm, followed by enhancements in patch-based point cloud compression [36]. On another front, the authors of [24] attempted to process coordinates directly based on PointNet++ [37]. While this method performed well on small and sparse models, its limited robustness became apparent, as global features used for reconstruction struggled to generate accurate contours on unseen shapes. Depoco [16] developed an autoencoder compression framework for sparse point cloud graphs, utilizing features extracted from the encoder and associated points as compression data. Liang et al. [17] introduced a novel network model called TransPCC, employing a Transformer autoencoder architecture for point cloud geometry compression. Despite its high robustness, its application remained confined to sparse points within blocks segmented from original dense point clouds. These point-based compression methods avoid the accuracy loss seen in voxel networks, but their complex network structures limit their ability to effectively utilize dense points. Octsqueeze [18] and VoxelcontextNet [38] are part of a series of works dedicated to outdoor scene compression, introducing network improvements in entropy coding performance. Muscle [39] took a step further by incorporating spatiotemporal relationships between multiple scans of outdoor scenes, thereby reducing the bitrate of both geometric and intensity values.

2.3. Transformer Methods

Transformer [40] is a model based on a self-attention mechanism that not only performs powerfully in modeling global contexts, but also shows excellent transferability to downstream tasks under large-scale pre-training. This success is widely witnessed in the fields of machine translation and natural language processing. In 2018, Devlin et al. [41]

proposed the Bert model based on a Transformer with a bidirectional coding structure based on a masking mechanism, achieving state-of-the-art results on multilingual tasks. In addition, many Transformer-based language models, including Bert, such as GPTv1-3 [42–44], T5 [45], etc., have demonstrated strong performance. CNNs have been dominant in computer vision tasks due to their inherent inductive preferences, such as translation invariance, localization, and other properties. However, the limited sensory field of CNNs makes it difficult to capture global contextual information. Inspired by the success of the Transformer model on linguistic tasks, several recent studies have applied Transformer to computer vision tasks. Parmar et al. [46] designed the image Transformer model for image generation tasks based on the problem of generating or transforming autoregressive sequences for the Transformer decoder. Recently, Dosovitskiy et al. [47] introduced another visual Transformer model, ViT, which adopts the standard structure of Transformer in its structure completely. ViT has achieved state-of-the-art performance on several image recognition benchmark tasks. Point Transformer V2 [48] introduced grouped vector attention along with a weight encoding layer, expanding upon the Point Transformer [49] architecture. Moreover, the Transformer architecture has found applications in numerous other computer vision challenges, including target detection [50–52] and semantic segmentation [53–56]. Owing to their outstanding performance, Transformer-based models are being adopted by an increasing number of researchers to enhance various vision-related tasks.

3. Methods

The network architecture for point cloud geometry compression based on Transformer, known as TransPCGC, is depicted in Figure 1. It encompasses a preprocessing module, a compression network, an entropy coding module, and a post-processing module.

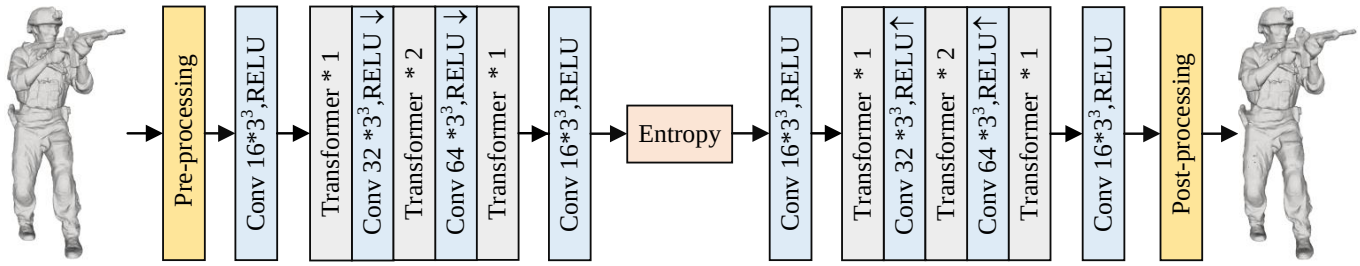

Figure 1. TransPCGC. "Conv" denotes the number of output channels, and the kernel size, "Transformer *1/*2", indicates the cascading of one/two Transformer modules. "↑" and "↓" signify the operations of zoom-in and zoom-out, "ReLU" refers to the rectified linear unit, Entropy is expressed as Entropy Rate Modeling.

3.1. Pre-Processing

The original point cloud's representation format preserves discrete geometric information in a three-dimensional space. However, this format cannot be directly utilized for 3D convolutions. Therefore, the TransPCGC method initially processes the original point cloud representation, converting it from the discrete point set format to a volumetric model represented by voxels. Voxelized point clouds describe correlations between voxels in a three-dimensional space. This aids deep learning techniques in extracting meaningful features from point clouds. The voxel representation format has a unit length of 1, and when scaling or downsampling is necessary, the voxel size is typically set to a value greater than 1. End-to-end neural network algorithms depend on the backpropagation function. To ensure differentiability, a method involving the addition of stochastic quantization noise is employed. This allows gradients propagation and facilitation of updates to model parameters during the training process, as illustrated in Equation (1). Here, t represents the

quantized data, μ represents random uniform noise with a distribution range from $-\frac{1}{2}$ to $\frac{1}{2}$, and $\hat{t}$ represents the simulated quantized data after noise addition.

$$\hat{t} = t + \mu. \tag{1}$$

The cost of processing a whole voxel rises dramatically when voxels are used, leading to a large increase in data volume. The processing cost can be decreased by dividing the voxels into non-overlapping chunks. These blocks are typically of a smaller size, denoted by $M \times M \times M$, as expressed in Equation (2). Here, (b_c, n_c, g_c) represents the specified position of a block, $\hat{T}_n$ denotes the global coordinates of the voxel, and $\hat{T}_n^{loc}$ signifies the local coordinates of all blocks (b_c, n_c, g_c) encoded using an octree encoder.

$$\hat{T}_n^{loc} = \hat{T}_n - (b_c \times M, n_c \times M, g_c \times M). \tag{2}$$

3.2. Encoder and Decoder Modules

The encoder–decoder of TransPCGC primarily consists of Transformer blocks, which extract both global information and local neighborhood features from the point cloud. As the decoder shares the same module composition as the encoder, it is introduced in the following subsection together.

Transformer Block

The transformer encoder consists of two components. One is the attention module for mixing information among tokens and we term it a token mixer. The other component contains the remaining modules, such as channel MLPs and residual connections. The computational complexity of multi-head attention increases quadratically with the number of tokens [57]. In related work, Ref. [58], attention was replaced with Fourier Transform and still achieved an around 97% accuracy compared to vanilla Transformers. Inspired by [59], we introduced a pooling operation that does not require any learnable parameters as a token mixer to compute long-distance dependencies. This approach ensures global feature extraction while significantly reducing both computations and parameters. Additionally, we utilized convolutional layers for feature extraction. These convolutional layers preserve the spatial structure of the point cloud, and the parameter count is independent of the input point cloud's size, leading to a further reduction in parameters. Specifically, we used a convolution kernel size of $1 \times 1 \times 1$. Consequently, our developed Transformer module excels at extracting local features with translation invariance using convolution operations and capturing long-distance information in sequences. We applied the proposed Transformer block to compress point cloud geometry data, resulting in promising experimental outcomes. Figure 2 illustrates the schematic diagram of the structure.

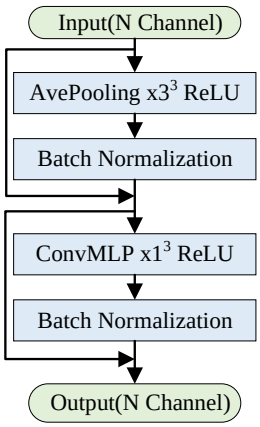

Figure 2. Transformer block.

First, the input $\hat{T}_n^{loc}$ undergoes an input embedding process [60], where it is partitioned into patches and transformed into a token sequence, as demonstrated in Equation (3). Then, the tokens are processed by the Token Mixer within the Transformer, enabling the exchange of information between tokens, as depicted in Equation (4).

$$Q = \text{InputEmb}\left(\hat{T}_n^{loc}\right), \tag{3}$$

$$G = \text{TokenMixer}(\text{Norm}(Q)) + Q. \tag{4}$$

Norm(·) represents normalization techniques such as Group Normalization, Layer Normalization, and Batch Normalization. TokenMixer(·) is a token mixer constructed using a pooling operation without any learnable parameters. ConvMLP consists of a convolution operation with a kernel size of $1 \times 1 \times 1$, as shown in Equation (5). W_1 and W_2 represent the learnable parameters within ConvMLP(), and they are accompanied by a nonlinear activation function such as GELU, ReLU, or Swish.

$$Z = \sigma(\text{Norm}(G)W_1)W_2 + G. \tag{5}$$

3.3. Entropy Coding

As a result of its superior compression capabilities, arithmetic coding is frequently used in entropy encoding, and TransPCGC also employs it to compress features. To enable more accurate probability estimation within the neural network, Equation (6) is used to approximate the bitstream quantization of the features. Here, E represents the quantization process, and $p_{\hat{t}}$ stands for the probability density function of the input value. TransPCGC draws inspiration from feature encoders and hyperprior encoders in image compression to achieve conditional probability estimation. In the hyperprior encoder, convolutional operations are likewise utilized to extract hyperprior information z, which assists in decoding the hyperprior information for auxiliary bitstream estimation of $\hat{t}$. Quantization strategies are employed for manipulating the hidden feature t and hyperprior z. The entropy model represents a fully factorized probabilistic model, in Equation (7), where U represents a uniform distribution and $\psi^{(i)}$ denotes the parameters of each univariate distribution $p_{\hat{z}_i|\psi^{(i)}}$. As for $\hat{t}$, it is assumed to follow a Laplace distribution L, and the probability density function is estimated based on the hyperprior $\hat{z}$, in Equation (8). Here, the hyperprior $\hat{z}$ yields the variance σ_i, mean μ_i, and parameters for each element of $\hat{t}_i$.

$$R_{\hat{t}} = E\left[-\log_2 p_{\hat{t}}(\hat{t})\right], \tag{6}$$

$$p_{\hat{z}|\psi}(\hat{z} \mid \psi) = \prod\left(p_{\hat{z}_i|\psi^{(i)}}\left(\psi^{(i)}\right) * U(-1/2, 1/2)\right)(\hat{z}_i), \tag{7}$$

$$p_{\hat{t}|\hat{z}}(\hat{t} \mid \hat{z}) = \prod_i (\mathcal{L}(\mu_i, \sigma_i) * U(-1/2, 1/2))\left(\hat{t}_i\right). \tag{8}$$

3.4. Loss Function

It is common practice to consider both the bitstream and the distortion simultaneously while optimizing the compression of point clouds, utilizing the Lagrangian loss and weighted binary cross-entropy as a joint loss function. In Equation (9), this optimization strategy balances the bitstream and distortion tradeoff. Here, the bitstream is symbolized by R, the distortion is represented by D, and λ regulates the weight of the BD-rate. As demonstrated in Equation (10), the bitstream is formed from t_i and z_i, and its estimation can be stated using the cross-entropy loss function. The two bitstreams are added to create the complete bitstream, which is represented as $R = R_{\hat{t}} + R_{\hat{z}}$, with the hyperprior bitstream $\hat{z}$ acting as auxiliary data.

$$J_{loss} = R + \lambda D, \tag{9}$$

$$\begin{cases} R_{\hat{t}} = \sum_i - \log_2\left(p_{\hat{t}_i|\hat{z}_i}\left(\hat{t}_i \mid \hat{z}_i\right)\right) \\ R_{\hat{z}} = \sum_i - \log_2\left(p_{\hat{z}_i|\psi^{(i)}}\left(\hat{z}_i \mid \psi^{(i)}\right)\right) \end{cases}. \tag{10}$$

To evaluate the distortion of point cloud compression, the weighted binary cross-entropy loss function is employed to classify the decoded voxels. This loss function, as depicted in Equation (11), measures the probability value, denoted as $p_{\tilde{x}} = sigmoid(\tilde{x})$, of voxel occupancy ranging between 0 and 1. Here, $\tilde{x}_o$ represents the occupied voxel, $\tilde{x}_n$ signifies the unoccupied voxel, and α is a hyperparameter that controls the relative weight, typically set to 3. Furthermore, N_o and N_n represent the quantities of occupied and unoccupied voxels, respectively.

$$D_{WBCE} = \frac{1}{N_o}\sum^{N_o} - \log p_{\tilde{x}_o} + \alpha\frac{1}{N_n}\sum^{N_n} - \log(1 - p_{\tilde{x}_n}). \tag{11}$$

3.5. Post-Processing

As the decoded voxels $\tilde{x}$ fall within the interval of (0, 1), it becomes necessary to perform a binary classification operation on the decoded voxels. In TransPCGC, an adaptive thresholding approach is employed, where the threshold is determined based on the number of occupied voxels in the original point cloud. Subsequently, the position information of points is extracted from the voxel blocks.

4. Experiment

4.1. Experimental Set

We randomly selected approximately 12,000 point cloud models from ShapeNet [61] for our study. During training, these point clouds were randomly cropped into blocks of size $64 \times 64 \times 64$, resulting in a total of 3×10^5 blocks. The loss function employed in our approach is presented in Equation (9). To generate models with different compression rates, we set the rate-distortion weight λ to 2, 3.5, 6, 10, and 16. During the training process, we initially set λ to 16 and trained a high-rate model, which was then utilized to initialize the training of low-rate models. We used a learning rate of 10^{-5}, a batch size of eight, and the AdamW optimizer. After 2×10^5 iterations, the training models began to converge. The experiments were conducted using the TensorFlow 2.4 deep learning framework on the Ubuntu 20.04 platform. The system configuration included an Intel Xeon(R) Gold 6134 processor (Intel Corporation, Santa Clara, CA, USA), 256 GB of memory, and an NVIDIA GeForce RTX 3090 GPU (Nvidia Corporation, Santa Clara, CA, USA).

4.2. Experiment Results

The experiment evaluated the compression performance of TransPCGC using five datasets: 8i Voxelized FullBodies (8iVFB) [62], Owlii dynamic human mesh (Owlii) [63], MUVB [64], ShapeNet-Core [61], and ModelNet40 [65]. Because of the substantial size of the ShapeNet-Core [61] and ModelNet40 [65] datasets, 32 mesh models were randomly selected from each dataset and subjected to random rotations and voxelization.

The proposed TransPCGC was compared and analyzed against other point cloud geometry compression algorithms, including G-PCC (O) [20], G-PCC (T) [20], GeoCNNv2 [6], ADL-PCC [14], Learned-PCGC [7], MRM-PCGC [33] and VA-PCC [34]. Among them, MRM-PCGC [33] and VA-PCC [34] utilize data from their papers for comparison. Objective comparisons were performed using BD-rate analysis for lossy compression, measuring point-to-point error (D1), point-to-plane error (D2), and PSNR derived from both metrics. Additionally, the bits per input point (bpp) were used to assess the compression rate. Figure 3 shows the rate-distortion curves for different point clouds. It can be observed that the D1 and D2 metrics of TransPCGC outperform the compared algorithms across different bitrates. The results are shown in Tables 1 and 2. Our method achieved averages −80.84% and −77.59% gains against G-PCC(T) [20], −97.09% and −97.37% gains against

G-PCC(O) [20], −37.9% and −40.27% gains against ADL-PCC [14], −39.01% and −34.64% gains against GeoCNNv2 [6], −14.39% and −16.66% gains against Learned-PCGC [7], −4.63% and −10.78% gains against MRM-PCGC [33], −14.12% and −15.99% gains against VA-PCGC [34] measured via respective D1 and D2 based BD-Rate.

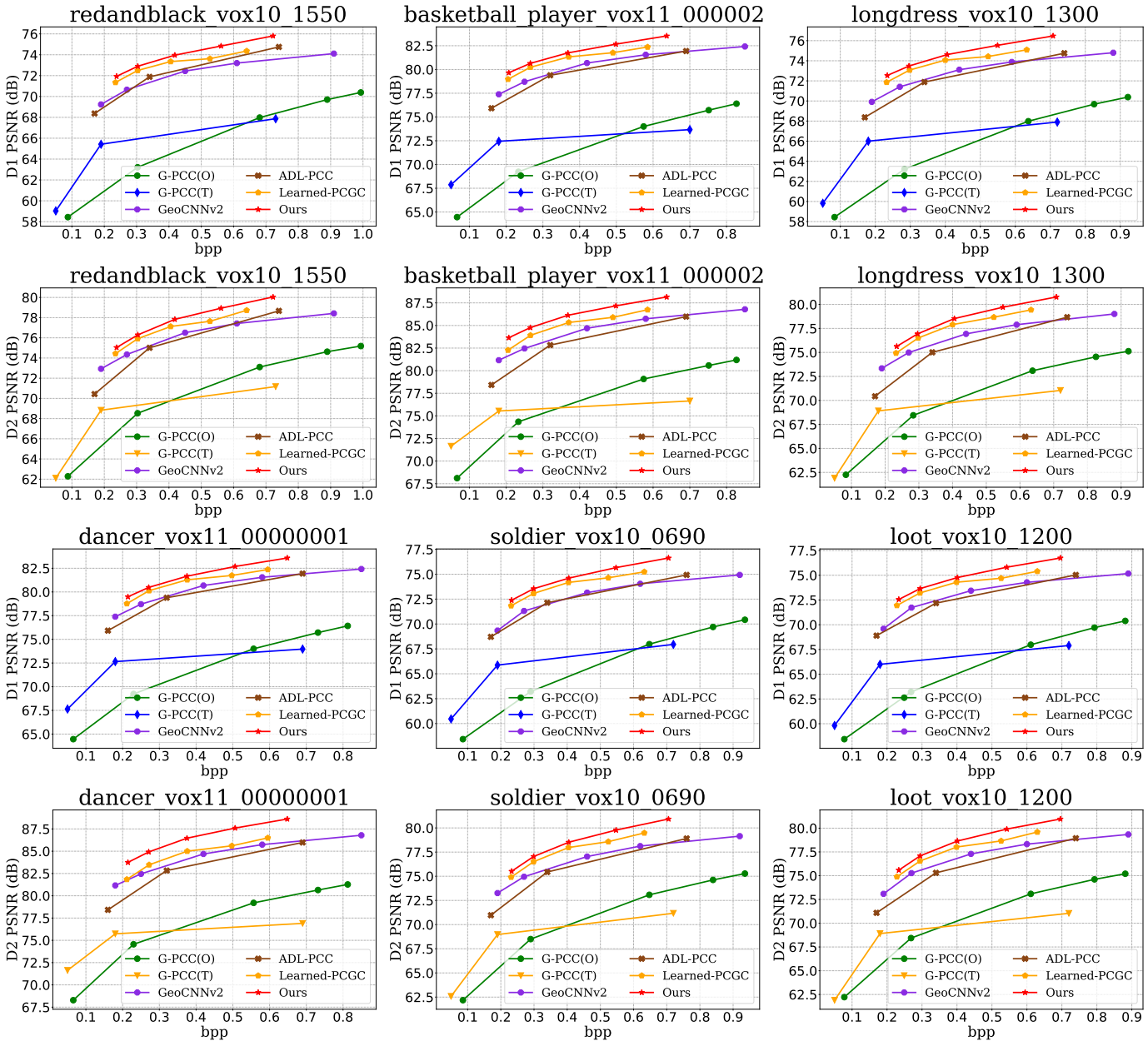

Figure 3. Rate distortion curves plotted on different test samples.

Table 1. BD-rate reduced G-PCC(T) [20], G-PCC(O) [20], ADL-PCC [14], GeoCNNv2 [6], Learned-PCGC [7], MRM-PCGC [33] and VA-PCGC [34] in D1 based BD-rate measurement (percentage/%).

Point Clouds	D1(p2point)						
	G-PCC(T) [20]	G-PCC(O) [20]	ADL-PCC [14]	GeoCNNv2 [6]	Learned-PCGC [7]	MRM-PCGC [33]	VA-PCC [34]
Longderss_vox10	−81.35	−97.33	−42.36	−43.08	−16.31	−5.98	−15.63
Loot_vox10	−79.47	−96.96	−40.42	−39.5	−14.9	−5.74	−15.06
Red&black_vox10	−80.71	−92.04	−29.52	−44.58	−17.26	−1.68	−10.46
Soldier_vox10	−80.62	−95.48	−39.4	−43.42	−41.21	−5.11	−13.47
Queen_vox10	−79.19	−98.23	−34.1	−29.07	−9.73	-	-
Player_vox11	−81.76	−99.85	−41.63	−38.55	−15.07	-	−15.29
Dancer_vox11	−82.79	−99.71	−37.89	−34.89	−13.24	-	−14.78
Average	−80.84	−97.09	−37.9	−39.01	−14.39	−4.63	−14.12

Table 2. BD-rate reduced G-PCC(T) [20], G-PCC(O) [20], ADL-PCC [14], GeoCNNv2 [6], Learned-PCGC [7], MRM-PCGC [33] and VA-PCGC [34] in D2 based BD-rate measurement (percentage/%).

Point Clouds	D2(p2plane)						
	G-PCC(T) [20]	G-PCC(O) [20]	ADL-PCC [14]	GeoCNNv2 [6]	Learned-PCGC [7]	MRM-PCGC [33]	VA-PCC [34]
Longderss_vox10	−76.56	−96.95	−41.07	−36.26	−12.24	−12.19	−17.79
Loot_vox10	−74.84	−96.41	−39.5	−32.37	−13.53	−11.22	−16.37
Red&black_vox10	−75.28	−94.24	−30.47	−33.16	−13.17	−9.21	−14.99
Soldier_vox10	−75.97	−95.25	−37.8	−36.57	−12.95	−10.51	−15.82
Queen_vox10	−75.27	−98.78	−38.07	−20.81	−11.78	-	-
Player_vox11	−82.66	−99.99	−47.06	−40.72	−21.62	-	−15.48
Dancer_vox11	−82.52	−99.98	−47.95	−42.58	−31.3	-	−15.51
Average	−77.59	−97.37	−40.27	−34.64	−16.66	−10.78	−15.99

From Table 3, it can be observed that our method, TransPCGC, demonstrates superior performance when compared to the G-PCC(O) [20] method on the 8iVFB, Owlii, and MUVB datasets. Across the three datasets, our method achieves an average BD-rate gain of 85.79% and 80.24% for D1 and D2 metrics, respectively. We also compared our proposed TransPCGC method with the Learned-PCGC [7] method on five datasets. Across all five datasets, our method showcases average BD-rate gains of 18.26% and 13.83% for D1 and D2 metrics, respectively. These results demonstrate the superior performance of our method compared to the comparative approaches across multiple datasets.

Table 3. BD-rate reducest G-PCC(O) [20] and Learned-PCGC [7] in D1 and D2 based BD-rate measurement (percentage/%).

Point Clouds	G-PCC(O) [20]		Learned-PCGC [7]	
	D1	D2	D1	D2
8iVFB	−96	−96.32	−15.13	−13.08
Owlii	−78.03	−75.55	−13.45	−29.4
MVUB	−83.34	−68.85	−9.86	−2.53
ModelNet	-	-	−24.57	−12.06
ShapeNet	-	-	−28.27	−12.1
Average	−85.79	−80.24	−18.26	−13.83

In Figures 4 and 5, we provide a visual comparison between the original point cloud and the decompressed point clouds. To visualize the geometric details of the point cloud, we initially compute the normal vectors for each point using the surrounding 20 neighboring points and render the point cloud. This approach provides a superior visual effect.

Additionally, we generate error maps based on the D1 error between the decompressed and original point cloud, allowing us visualization of the error distribution. Comparing the resulting point clouds from different compression methods, our method stands out in preserving intricate details and generating visually superior point clouds, as indicated by the red and blue dashed boxes in Figures 4 and 5. It is noteworthy that G-PCC (T) [20] introduces noticeable holes, while both G-PCC (O) [20] and G-PCC (T) [20] exhibit sparser reconstructed point clouds with significant quantization loss compared to other methods. This phenomenon occurs because octree-based encoding methods often lose details in leaf nodes at low bit rates. On the contrary, ADL-PCC [14] and Learned-PCC [7] demonstrate better preservation of details and fill a significant number of holes. However, our method surpasses them in terms of detail preservation.

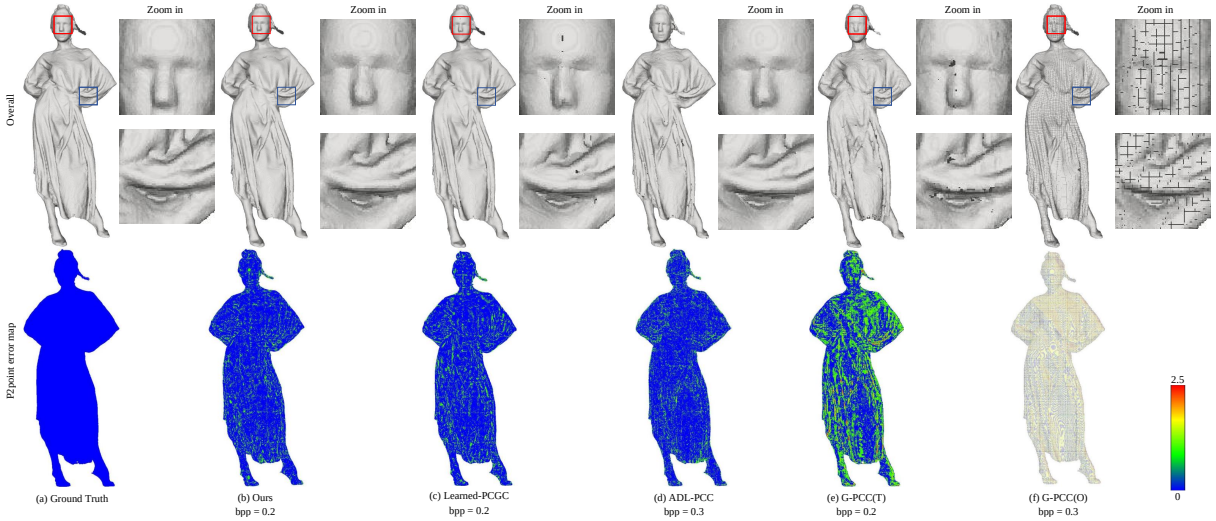

Figure 4. Visual comparison of "longdress" for Ground Truth, Ours, Learned-PCGC [7], ADL-PCC [14], G-PCC (T) [20] and G-PCC (O) [20].

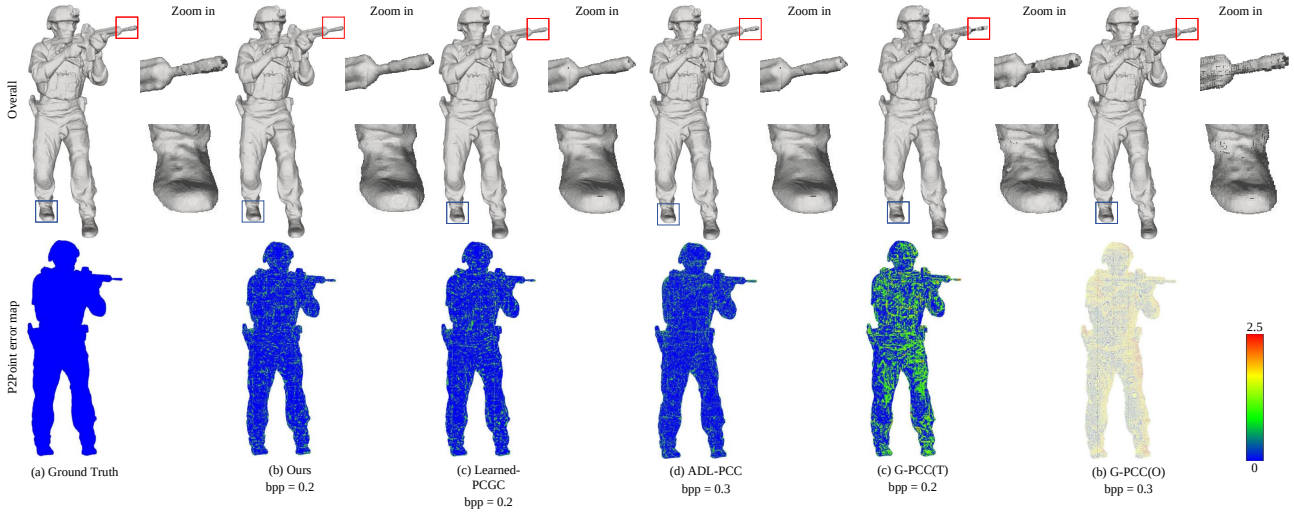

Figure 5. Visual comparison of "Soldier" for Ground Truth, Ours, Learned-PCGC [7], ADL-PCC [14], G-PCC (T) [20] and G-PCC (O) [20].

In Table 4, we compare the running time of different methods. The table displays the encoding and decoding time as well as the storage space of each method. It is evident that the traditional method G-PCC(O) [20] has the fastest encoding and decoding time, resulting in a 16% reduction in encoding and decoding time and a 50% reduction in storage space compared to the Learned-PCGC [7]. The slightly slower decoding time of our method is primarily attributed to the generation of more voxels in the reconstruction process, which

is based on a binary classification method. We plan to further optimize this aspect in future work.

Table 4. Compare the Enc time, Dec time, and Model size of different methods (unit: second/s).

	G-PCC(O) [20]	G-PCC(T) [20]	Learned-PCGC [7]	Ours
Encoding time	1.6	8.16	6.63	7.42
Decoding time	0.6	6.58	17.49	12.85
Model size	2.6 M	2.6 M	8.0 M	4.0 M

4.3. Ablation Study

The effectiveness of the Transformer modules in TransPCGC was validated through disintegration experiments. We performed disintegration by stacking and adjusting various parameters of the Transformer Blocks, including the network structure, activation function, normalization layer, and patch size. The outcomes indicated that utilizing ReLU activation, Batch Normalization, and a patch size of seven led to the designed Transformer Blocks achieving superior BD-Rate, as demonstrated in Table 5.

Table 5. Ablation for Transformer Blocks on the "longdress" point cloud.

Ablation	Variant	Bpp	D1	D2
Baseline	[Trans, Trans, Trans]	0.59	75.45	79.53
Hybrid Stages	[Trans, Trans, Trans] → [Trans, Trans*2, Trans]	0.58	75.48	79.57
	[Trans, Trans, Trans] → [Trans, Trans*3, Trans]	0.59	75.45	79.53
Activation	GeLU → ReLU	0.55	75.45	79.58
	GeLU → Swich	0.55	74.53	78.58
Normalization	Group Normalization → Layer Normalization	0.54	75.22	79.36
	Group Normalization→ Batch Normalization	0.55	75.54	79.71
Token mixers	Pooling size 7→5	0.55	75.47	79.62
	Pooling size 7→9	0.55	75.46	79.58

5. Conclusions

In this paper, we introduce a Transformer-based point cloud geometric compression network. It includes preprocessing modules for voxelization and partitioning of point clouds, compression networks to achieve optimal BD-Rate, and postprocessing modules for point cloud reconstruction. This approach addresses the common problem in mainstream point cloud geometric compression algorithms, which often overlook global information in point cloud data. We propose a low-complexity and low-computation Transformer module, incorporating a pooling operation that does not require any learnable parameters as a token mixer to compute long-distance dependencies. This method ensures global feature extraction while significantly reducing both computations and parameters. In addition, we employ the convolutional layer for feature extraction, which further reduces the number of parameters while preserving the spatial structure of the point cloud. The experimental results demonstrate that our method improves the reconstruction quality of the point cloud and preserves more details. Moreover, our method outperforms other compared algorithms in terms of BD rate gain. Compared with the data from the latest VA-PCC published paper, our method achieves an average BD-Rate gain of 14.12% and 15.99% for D1 and D2 distortion criteria, respectively. Furthermore, when compared with the Learned-PCGC network, our method shows improvement in BD-Rate while reducing encoding and decoding time by 16% and model size by 50%. There are numerous interesting topics for exploration in the future, including (1) enhancing the entropy coding module by constructing more accurate context probability models using the Transformer module, (2) attribute compression (e.g., RGB colors), and (3) motion capturing for dynamic point clouds.

Author Contributions: Conceptualization, S.L. and H.Y.; data curation, H.Y. and C.H.; formal analysis, S.L.; funding acquisition, H.Y. and C.H.; investigation, S.L.; methodology, S.L. and C.H.; project administration, H.Y. and C.H.; resources, H.Y. and C.H.; writing—original draft, S.L.; writing—review and editing, S.L. and C.H. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been partially supported by the National Key R&D Program of China under grant No. 2020YFB1709200.

Data Availability Statement: The dataset used in this paper is the publicly available shapenet, ModelNet40, MVUB, 8iVFB, and Owlii. They can be downloaded at the following links: https://shapenet.org/, https://modelnet.cs.princeton.edu/, https://plenodb.jpeg.org/ and https://mpeg-pcc.org/index.php/pcc-content-database/ (accessed on 15 October 2023).

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

References

1. Zhu, Z.; Peng, S.; Larsson, V.; Xu, W.; Bao, H.; Cui, Z.; Oswald, M.R.; Pollefeys, M. Nice-slam: Neural implicit scalable encoding for slam. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 12786–12796.
2. Li, Y.; Zhao, Z.; Fan, J.; Li, W. ADR-MVSNet: A cascade network for 3D point cloud reconstruction with pixel occlusion. *Pattern Recognit.* **2022**, *125*, 108516. [CrossRef]
3. Fernandes, D.; Silva, A.; Névoa, R.; Simões, C.; Gonzalez, D.; Guevara, M.; Novais, P.; Monteiro, J.; Melo-Pinto, P. Point-cloud based 3D object detection and classification methods for self-driving applications: A survey and taxonomy. *Inf. Fusion* **2021**, *68*, 161–191. [CrossRef]
4. Vujasinović, M.; Regodić, M.; Kecman, S. Point cloud processing software solutions. *AGG+ J. Archit. Civ. Eng. Geod. Relat. Sci. Fields* **2020**, *8*, 64–75. [CrossRef]
5. 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), Taipei, Taiwan, 22–25 September 2019; pp. 4320–4324. [CrossRef]
6. 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), Tampere, Finland, 21–24 September 2020; pp. 1–6. [CrossRef]
7. Wang, J.; Zhu, H.; Liu, H.; Ma, Z. Lossy Point Cloud Geometry Compression via End-to-End Learning. *IEEE Trans. Circuits Syst. Video Technol.* **2021**, *31*, 4909–4923. [CrossRef]
8. Wang, J.; Ding, D.; Li, Z.; Ma, Z. Multiscale Point Cloud Geometry Compression. In Proceedings of the 2021 Data Compression Conference (DCC), Snowbird, UT, USA, 23–26 March 2021; pp. 73–82. [CrossRef]
9. Wang, J.; Ding, D.; Li, Z.; Feng, X.; Cao, C.; Ma, Z. Sparse Tensor-Based Multiscale Representation for Point Cloud Geometry Compression. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 9055–9071. [CrossRef] [PubMed]
10. Nguyen, D.T.; Quach, M.; Valenzise, G.; Duhamel, P. Learning-Based Lossless Compression of 3D Point Cloud Geometry. In Proceedings of the 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Toronto, ON, Canada, 6–11 June 2021; pp. 4220–4224. [CrossRef]
11. Nguyen, D.T.; Quach, M.; Valenzise, G.; Duhamel, P. Multiscale deep context modeling for lossless point cloud geometry compression. In Proceedings of the 2021 IEEE International Conference on Multimedia & Expo Workshops (ICMEW), Shenzhen, China, 5–9 July 2021; pp. 1–6. [CrossRef]
12. Nguyen, D.T.; Quach, M.; Valenzise, G.; Duhamel, P. Lossless Coding of Point Cloud Geometry Using a Deep Generative Model. *IEEE Trans. Circuits Syst. Video Technol.* **2021**, *31*, 4617–4629. [CrossRef]
13. Nguyen, D.T.; Kaup, A. Lossless Point Cloud Geometry and Attribute Compression Using a Learned Conditional Probability Model. *IEEE Trans. Circuits Syst. Video Technol.* **2023**, *33*, 4337–4348. [CrossRef]
14. Guarda, A.F.R.; Rodrigues, N.M.M.; Pereira, F. Adaptive Deep Learning-Based Point Cloud Geometry Coding. *IEEE J. Sel. Top. Signal Process.* **2021**, *15*, 415–430. [CrossRef]
15. Wang, J.; Ding, D.; Ma, Z. Lossless Point Cloud Attribute Compression Using Cross-scale, Cross-group, and Cross-color Prediction. In Proceedings of the 2023 Data Compression Conference (DCC), IEEE, Snowbird, UT, USA, 21–24 March 2023; pp. 228–237.
16. Wiesmann, L.; Milioto, A.; Chen, X.; Stachniss, C.; Behley, J. Deep Compression for Dense Point Cloud Maps. *IEEE Robot. Autom. Lett.* **2021**, *6*, 2060–2067. [CrossRef]
17. Liang, Z.; Liang, F. TransPCC: Towards Deep Point Cloud Compression via Transformers. In Proceedings of the 2022 International Conference on Multimedia Retrieval, Newark, NJ, USA, 27–30 June 2022; pp. 1–5.
18. Huang, L.; Wang, S.; Wong, K.; Liu, J.; Urtasun, R. Octsqueeze: Octree-structured entropy model for lidar compression. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 1313–1323.

19. Beemelmanns, T.; Tao, Y.; Lampe, B.; Reiher, L.; van Kempen, R.; Woopen, T.; Eckstein, L. 3D Point Cloud Compression with Recurrent Neural Network and Image Compression Methods. In Proceedings of the 2022 IEEE Intelligent Vehicles Symposium (IV), IEEE, Aachen, Germany, 5–9 June 2022; pp. 345–351.
20. *ISO/IEC 23090-9:2023*; Information Technology-Coded Representation of Immersive Media-Part 9: Geometry-Based Point Cloud Compression (G-PCC). MPEG: Kowloon, Hong Kong, 2023.
21. *ISO/IEC 23090-5:2021*; Information Technology-Coded Representation of Immersive Media-Part 5: Visual Volumetric Vedio-Based Coding (V3C) and Vedio-Based Point Cloud Compression (V-PCC). MPEG: Kowloon, Hong Kong, 2021.
22. Dvořák, J.; Káčereková, Z.; Vaněček, P.; Váša, L. Priority-based encoding of triangle mesh connectivity for a known geometry. *Comput. Graph. Forum* **2023**, *42*, 60–71. [CrossRef]
23. 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.
24. Huang, T.; Liu, Y. 3d point cloud geometry compression on deep learning. In Proceedings of the 27th ACM International Conference on Multimedia, Nice, France, 21–25 October 2019; pp. 890–898.
25. Google. Draco 3D Data Compression. 2017. Available online: https://github.com/google/draco (accessed on 15 October 2023).
26. Dumic, E.; Bjelopera, A.; Nüchter, A. Dynamic point cloud compression based on projections, surface reconstruction and video compression. *Sensors* **2021**, *22*, 197. [CrossRef] [PubMed]
27. Yu, S.; Sun, S.; Yan, W.; Liu, G.; Li, X. A method based on curvature and hierarchical strategy for dynamic point cloud compression in augmented and virtual reality system. *Sensors* **2022**, *22*, 1262. [CrossRef] [PubMed]
28. 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] [PubMed]
29. Puang, E.Y.; Zhang, H.; Zhu, H.; Jing, W. Hierarchical Point Cloud Encoding and Decoding with Lightweight Self-Attention Based Model. *IEEE Robot. Autom. Lett.* **2022**, *7*, 4542–4549. [CrossRef]
30. Tatarchenko, M.; Dosovitskiy, A.; Brox, T. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2088–2096.
31. Dai, A.; Qi, C.R.; Nießner, M. Shape Completion Using 3D-Encoder-Predictor CNNs and Shape Synthesis. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 6545–6554. [CrossRef]
32. Luo, S.; Hu, W. Diffusion probabilistic models for 3d point cloud generation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 2837–2845.
33. Yu, J.; Wang, J.; Sun, L.; Wu, M.E.; Zhu, Q. Point Cloud Geometry Compression Based on Multi-Layer Residual Structure. *Entropy* **2022**, *24*, 1677. [CrossRef] [PubMed]
34. Zhuang, L.; Tian, J.; Zhang, Y.; Fang, Z. Variable Rate Point Cloud Geometry Compression Method. *Sensors* **2023**, *23*, 5474. [CrossRef] [PubMed]
35. You, K.; Gao, P. Patch-based deep autoencoder for point cloud geometry compression. *arXiv* **2021**, arXiv:2110.09109.
36. You, K.; Gao, P.; Li, Q. IPDAE: Improved Patch-Based Deep Autoencoder for Lossy Point Cloud Geometry Compression. In Proceedings of the 1st International Workshop on Advances in Point Cloud Compression, Processing and Analysis, Lisbon, Portugal, 14 October 2022; pp. 1–10.
37. Qi, C.R.; Yi, L.; Su, H.; Guibas, L.J. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In Proceedings of the Advances in Neural Information Processing Systems (NIPS), Long Beach, CA, USA, 4–9 December 2017; pp. 5099–5108.
38. Que, Z.; Lu, G.; Xu, D. Voxelcontext-net: An octree based framework for point cloud compression. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 6042–6051.
39. Biswas, S.; Liu, J.; Wong, K.; Wang, S.; Urtasun, R. Muscle: Multi sweep compression of lidar using deep entropy models. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 22170–22181.
40. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS), Long Beach, CA, USA, 4–9 December 2017; pp. 6000–6010.
41. Devlin, J.; Chang, M.-W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the naacL-HLT, Minneapolis, MN, USA, 2 June 2019; Volume 1, p. 2.
42. Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I. Improving Language Understanding by Generative Pre-Training. Technical Report, OpenAI. 2018. Available online: https://openai.com/research/language-unsupervised (accessed on 15 October 2023).
43. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language models are unsupervised multitask learners. *OpenAI Blog* **2019**, *1*, 9.
44. Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language models are few-shot learners. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 1877–1901.
45. Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P.J. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* **2020**, *21*, 5485–5551.

46. Parmar, N.; Vaswani, A.; Uszkoreit, J.; Kaiser, L.; Shazeer, N.; Ku, A.; Tran, D. Image transformer. In Proceedings of the International Conference on Machine Learning, PMLR, Stockholm, Sweden, 10–15 July 2018; pp. 4055–4064.
47. Kolesnikov, A.; Dosovitskiy, A.; Weissenborn, D.; Heigold, G.; Uszkoreit, J.; Beyer, L.; Minderer, M.; Dehghani, M.; Houlsby, N.; Gelly, S.; et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *arXiv* **2020**, arXiv:2010.11929.
48. Wu, X.; Lao, Y.; Jiang, L.; Liu, X.; Zhao, H. Point transformer v2: Grouped vector attention and partition-based pooling. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 33330–33342.
49. Zhao, H.; Jiang, L.; Jia, J.; Torr, P.H.; Koltun, V. Point transformer. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 16259–16268.
50. Misra, I.; Girdhar, R.; Joulin, A. An end-to-end transformer model for 3d object detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 2906–2917.
51. Sheng, H.; Cai, S.; Liu, Y.; Deng, B.; Huang, J.; Hua, X.S.; Zhao, M.J. Improving 3d object detection with channel-wise transformer. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 2743–2752.
52. Wang, Y.; Ye, T.; Cao, L.; Huang, W.; Sun, F.; He, F.; Tao, D. Bridged transformer for vision and point cloud 3d object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 12114–12123.
53. Gao, Y.; Liu, X.; Li, J.; Fang, Z.; Jiang, X.; Huq, K.M.S. LFT-Net: Local feature transformer network for point clouds analysis. *IEEE Trans. Intell. Transp. Syst.* **2022**, *24*, 2158–2168. [CrossRef]
54. Park, C.; Jeong, Y.; Cho, M.; Park, J. Fast point transformer. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 16949–16958.
55. Lai, X.; Liu, J.; Jiang, L.; Wang, L.; Zhao, H.; Liu, S.; Qi, X.; Jia, J. Stratified transformer for 3d point cloud segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 8500–8509.
56. Xu, S.; Wan, R.; Ye, M.; Zou, X.; Cao, T. Sparse cross-scale attention network for efficient lidar panoptic segmentation. In Proceedings of the AAAI Conference on Artificial Intelligence, Palo Alto, CA, USA, 22 February–1 March 2022; Volume 36, pp. 2920–2928.
57. Yu, W.; Luo, M.; Zhou, P.; Si, C.; Zhou, Y.; Wang, X.; Feng, J.; Yan, S. Metaformer is actually what you need for vision. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 10819–10829.
58. Lee-Thorp, J.; Ainslie, J.; Eckstein, I.; Ontanon, S. FNet: Mixing Tokens with Fourier Transforms. In Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Seattle, WA, USA, 10–15 July 2022; pp. 4296–4313.
59. Wang, W.; Xie, E.; Li, X.; Fan, D.P.; Song, K.; Liang, D.; Lu, T.; Luo, P.; Shao, L. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 568–578.
60. Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 10012–10022.
61. 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, 7–15 June 2015; pp. 1912–1920.
62. D'Eon, E.; Harrison, B.; Myers, T.; Chou, P.A. 8i voxelized full bodies—A voxelized point cloud dataset. *ISO/IEC JTC1/SC29 Jt.* **2017**, *7*, 11.
63. Xu, Y.; Lu, Y.; Wen, Z. Owlii Dynamic human mesh sequence dataset. In Proceedings of the ISO/IEC JTC1/SC29/WG11 m41658, 120th MPEG Meeting, Macau, 23–27 October 2017; Volume 1.
64. Loop, C.; Cai, Q.; Escolano, S.O.; Chou, P.A. Microsoft voxelized upper bodies—A voxelized point cloud dataset. In Proceedings of the ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) m38673/M72012, Geneva, Switzerland, 30 May–3 June 2016; Volume 1.
65. Chang, A.X.; Funkhouser, T.; Guibas, L.; Hanrahan, P.; Huang, Q.; Li, Z.; Savarese, S.; Savva, M.; Song, S.; Su, H.; et al. Shapenet: An information-rich 3d model repository. *arXiv* **2015**, arXiv:1512.03012.