Multiscale Latent-Guided Entropy Model for LiDAR Point Cloud Compression

Tingyu Fan, Linyao Gao, Yiling Xu, Member, IEEE, Dong Wang, Member, IEEE, and Zhu Li, Senior Member, IEEE

Abstract—The non-uniform distribution and extremely sparse nature of the LiDAR point cloud (LPC) bring significant challenges to its high-efficient compression. This paper proposes a novel end-to-end, fully-factorized deep framework that represents the original LiDAR point cloud into an octree structure and hierarchically constructs the octree entropy model in layers. The proposed framework utilizes a hierarchical latent variable as side information to encapsulate the sibling and ancestor dependence, which provides sufficient context information for the modeling of point cloud distribution while enabling the parallel encoding and decoding of octree nodes in the same layer. Besides, we propose a residual coding framework for the compression of the latent variable, which explores the spatial correlation of each layer by progressive downsampling, and model the corresponding residual with a fully-factorized entropy model. Furthermore, we propose soft addition and subtraction for residual coding to improve network flexibility. The comprehensive experiment results on the LiDAR benchmark SemanticKITTI and MPEG-specified dataset Ford demonstrate that our proposed framework achieves state-of-the-art performance among all the previous LPC frameworks. Besides, our end-to-end, fully-factorized framework is proved by experiment to be high-parallelized and time-efficient, which saves more than 99.8% of decoding time compared to previous state-of-the-art methods on LPC compression.

Index Terms—Point cloud compression, end-to-end learning, octree, entropy model, sparse convolution.

I. INTRODUCTION

Due to the rapid development of 3D sensors, point clouds (PCs) [1], [2], [3] have become a promising data structure with broad applications in autonomous driving [4], robotic sensing [5], and AR/VR. Among various use cases, a type of point cloud captured by the Light Detection And Ranging (LiDAR) sensor, called the LiDAR point cloud (LPC) [6], has recently received much attention. An LPC represents the scene information centered on a driving vehicle and is usually collected in real-time, with over 100,000 points per frame [7]. However, the non-uniform distribution of the LiDAR point cloud and the noise brought by real-time acquisition make the exploration of spatial correlation extremely difficult, bringing significant challenges to its compression. Therefore, this paper focuses on the geometry of LiDAR point clouds, aiming to design an efficient end-to-end deep framework for the compression of LiDAR point clouds.

Prior works on LiDAR point cloud compression use multiple data structures to regularize the non-uniform point cloud sequence, e.g., range images [8], KD-trees [9], [10] and Octrees [11], [12]. Among them, the Moving Picture Expert Group (MPEG) develops a Geometry-based Point Cloud Compression (G-PCC) standard [13]. G-PCC is implemented based on the octree structure that recursively divides the current 3D cube into eight sub-cubes. G-PCC reports state-of-the-art performance among all rule-based methods. However, the above methods are highly dependent on hand-crafted context selection and entropy model without the optimization by traversing the whole dataset, leading to unsatisfactory coding efficiency.

Thanks to the application of deep learning techniques in image/video compression [14], [15], [16], [17], [18], [19], learning-based point cloud compression methods [20], [21], [22], [23] have emerged. For LPC, most of the methods are based on the octree entropy model. Given the octree sequence \( x = \{x_1, x_2, \ldots, x_N\} \), the lower bound of bit rate can be formulated using the information entropy of the data distribution \( p(x) \) [24], which is assumed intractable due to its high dimensionality. Therefore, the goal is to approximate \( p(x) \) with an estimated distribution \( q(x) \) computed by a deep neural network. Given an estimation \( q(x) \), the lower bound of bit rate is the cross-entropy between \( q(x) \) and \( p(x) \), i.e.,

\[
H(p, q) = \mathbb{E}_{x \sim p} \left[ - \log q(x) \right],
\]

which is a tight lower bound achievable by arithmetic coding algorithms. Due to the high-dimensionality of \( x \), the probability model \( q(x) \) is often constructed in an auto-regressive [25], [26] fashion:

\[
q(x) = \prod_{i=1}^{N} q(x_i | \text{context}(x_i)),
\]

where \( \text{context}(x_i) \) is a subset of all decoded nodes that are considered to have dependence with \( x_i \). The strategy to determine the context set varies among different methods. OctSqueeze [20] assumes the conditional independence of sibling
nodes given their ancestors and applies a hierarchical coding structure that selects ancestry nodes as context, bringing low computational complexity at the cost of violating the original distribution. To mitigate the huge information brought by the sibling independence assumption, VoxelContext-Net [22] utilizes a 3D convolutional neural network (CNN) to capture the dependencies between nodes in the bottom layers of the octree. OctAttention [23] further encodes and decodes the octree in a breadth-first order and maintains a context window \( \{x_{i-N+1}, \ldots, x_{i-1}\} \) for attention-based context learning, which reports state-of-the-art performance among all LPC compression methods. However, selecting the decoded sibling nodes causes the context to vary during the decoding process. Thus the model needs to be constantly re-run to update the semantic information of the current context, leading to excessive decoding complexity and non-parallelism. Although the author proposes mask operation for complexity reduction, the model still requires even more than 10 minutes to decode one frame of point cloud [23].

To summarize, two major challenges must be compatibly addressed to design an efficient LPC compression network: 1) The context selection algorithm needs to thoroughly explore the dependence of the whole octree to avoid suboptimality. 2) The decoding process of nodes should be independent enough, such that the model runs only once during decoding for the sake of complexity and parallelism.

To this end, we propose a new end-to-end, fully-factorized LPC entropy model. Inspired by Ballé et al. [15], we address the above two issues by transmitting side information, which encodes additional information to reduce the mismatch of the entropy model. The proposed side information is a latent variable encapsulating the dependency information of the octree, through which the octree nodes are assumed to be independent layer-wisely. To fully explore the spatial correlation, we design the latent variable in a hierarchical form, progressively increasing the perceptive field to capture deeper correlation information. The model we propose encapsulates the sibling correlation through the latent variable to achieve optimal coding efficiency. Meanwhile, compared with Fu et al., the independence of sibling nodes conditioned on the latent variable significantly decreases the computational complexity and brings decoding parallelism among nodes in the same layer, such that the model runs only once during decoding.

To summarize, our contributions are highlighted as follows:

- We propose an end-to-end, fully-factorized LPC compression framework that encapsulates the spatial correlation of point clouds by introducing a hierarchical latent variable. The proposed framework explores the dependence of sibling nodes to avoid suboptimality while maintaining low complexity and a high degree of parallelism. The model runs only once during decoding.
- We propose to replace the ordinary addition/subtraction operation with soft addition/subtraction to improve the model’s flexibility.
- The experiment result shows that our proposed network achieves state-of-the-art performance on LPC datasets SemanticKitti [27] and Ford [28]. Furthermore, due to the enormous complexity reduction, our model saves 99.8% of runtime compared with the previous state-of-the-art method.

II. RELATED WORK

A. Rule-Based LiDAR Point Cloud Compression

Tree structures are widely applied in LPC compression due to its ability to regularize sparse geometry data, e.g., KD-tree [10], quadtree [29] and octree [11], [13]. Among them, the octree-based standard G-PCC [13] developed by the MPEG group achieves state-of-the-art performance. G-PCC relies on the decomposition of octree geometry. Specifically, a LiDAR point cloud \( P \) is first quantized into a \( 2^L \times 2^L \times 2^L \) cube:

\[
\hat{P} = \text{round} \left( \frac{P - \text{bias}}{qs} \right),
\]

\[
qs = \frac{\text{bounding}}{2^L - 1},
\]

where \( \text{bias} = \min(P_x), \min(P_y), \min(P_z) \), \( L \) is the target depth of the octree, and bounding is the size of the bounding box of \( P \). Correspondingly, the inverse quantization is defined as follows:

\[
\tilde{P} = \hat{P} * qs + \text{bias},
\]

where \( \tilde{P} \) is the reconstructed point cloud. As Figure 1 shows, the octree is constructed by recursively dividing the current cube of \( \hat{P} \) into 8 subcubes until the current cube is empty or the recursion reaches the target depth \( L \). An 8-bit occupancy code is assigned for each cube to indicate the occupancy situation (0 for unoccupied and 1 for occupied). However, the above methods (including G-PCC) rely on the hand-crafted context model that cannot be optimized by traversing large-scale data, leading to inferior performance compared with deep-learning based methods.

Besides tree structures, there are also attempts that transform a LiDAR point cloud into a “2.5D” depth image and leverage mature image/video codecs for its compression [30], [31], [32], [33], [34], [35], [36]. Among them, Tu et al. [30] convert LiDAR point clouds to range images and propose a framework for range image compression; Feng et al. [32] explore both spatial and temporal correlations to reduce redundancies while maintaining a low level of complexity; and Zhao et al. [35] propose Semantic Prior Representation (SPR) and an encoding algorithm for SPR information to improve the exploration of spatial correlation.

B. Learning-Based Point Cloud Compression

Due to the applications of deep learning techniques in image/video compression, learning-based point cloud compression methods have recently emerged. In general, these learning-based methods can be summarized as point-based, voxel-based and octree-based.

1) Point-Based Methods: The point-based methods directly process and compress the original point cloud, e.g., Huang et al. [37] utilizes PointNet++ [38] structure for feature extraction and point cloud reconstruction. Gao et al. [39] further propose graph-based feature extraction and attentive sampling to improve compression efficiency. However, due to
the complexity factor, the point-based methods are limited to small-scale datasets like ShapeNetCorev2 [40].

2) Voxel-Based Methods: The voxel-based methods process the point clouds by voxelization. Among them, Wang et al. [41] and utilize a 3D CNN-based network for voxelized point cloud compression. The experiment result shows remarkable coding gain; however, the voxelization process runs counter to the sparsity nature of point clouds, leading to excessive time and memory consumption. Therefore, Wang et al. further utilize the generalized sparse convolution by Choy et al. [42] and propose an end-to-end multi-scale point cloud compression framework based on the sparse CNN [43]. The experiment result shows state-of-the-art performance on dense point cloud datasets 8i Voxelized Full Bodies [44] (8iVFB) and Microsoft Voxelized Upper Bodies (MVUB) [45]. Apart from Wang et al. [41], [43], there are other attempts to use voxel-based frameworks for point cloud compression. For example, Guardar et al. [46] propose a PC block partitioning algorithm to ensure parallelism and spatial random access and compress each block using a 3D CNN with selectable modes. Quach et al. [47] introduce enhancements including deep analysis, flexible balancing weights into the focal loss and optimal decoding thresholding. However, the above voxel-based methods are limited to dense point clouds, which are not yet implementable on LPC [41], [43], [46], [47] due to the extreme sparsity and noise brought by real-time acquisition.

3) Octree-Based Methods: As mentioned in Section I, The octree-based methods utilize the octree structure to regularize the point cloud structure, which is widely used in LiDAR point cloud compression due to the ability to express sparse point clouds. The auto-regressive probability model is widely used to infer the high-dimensional octree sequence. However, due to the inefficiency of the auto-regressive inference, existing octree-based methods [20], [21], [22], [23] fail to balance the performance and complexity (more details in Section I).

III. PROBLEM FORMULATION

The proposed framework encodes and decodes the octree in layers to ensure parallelism. The octree is alternatively expressed as an ordered set of layers: \( x = \{x^{(1)} , x^{(2)} , \cdots , x^{(L)} \} \), where the decoding of the \( l \)-th layer \( x^{(l)} \) depends on the previously decoded layers, i.e., \( x^{(1 : l-1)} \). The corresponding hierarchical probability model is expressed as

\[
q(x) = \prod_{l=1}^{L} q(x^{(l)} | x^{(1 : l-1)}),
\]

The high dimensionality of \( x^{(l)} \) makes its probability distribution intractable to compute. Instead of relying on the inefficient auto-regressive models for the node-wise decomposition of \( x^{(l)} \), we introduce a latent variable \( f^{(l)} \), conditioned on which the nodes in \( x^{(l)} \) are assumed to be independent [48]. Figure 2 shows the decoding pipeline with the introduction of \( f^{(l)} \). Through \( f^{(l)} \), the spatial correlation of \( x^{(l)} \) is captured as follows:

\[
q(x^{(l)} | f^{(l)} , x^{(1 : l-1)}) = \prod_{i} q(x^{(l)}_{i} | f^{(l)} , x^{(1 : l-1)}) ,
\]

where \( N_{l} \) is the number of nodes in the \( l \)-th layer. \( f^{(l)} \) serves as side information to signal modifications to the probability distribution \( x^{(l)} \) that requires extra bits to express. Since \( f^{(l)} \) is to be encoded and decoded together with \( x^{(l)} \), we express the joint probability distribution of \( x^{(l)} \) and \( f^{(l)} \):

\[
q(x^{(l)} , f^{(l)} | x^{(1 : l-1)} , f^{(1 : l-1)} , \psi ) = q(x^{(l)} | x^{(1 : l-1)} , f^{(1 : l-1)} , \psi ) q(f^{(l)} | x^{(1 : l-1)} , f^{(1 : l-1)} , \psi ) ,
\]
where $\psi_x$ and $\psi_f$ are the parameters for decoding $x^{(l)}$ and $f^{(l)}$. By assuming the Markovian property, the joint distribution is reduced to the product of the following two terms:

$$
q_x^{(l)} = q(x^{(l)}|x^{(l-1)}, f^{(l)}, x^{(l-2)}, f^{(l-1)}; \psi_x),
$$

$$
q_f^{(l)} = q(f^{(l)}|x^{(l-1)}, f^{(l-1)}, x^{(l-2)}, f^{(l-2)}; \psi_f).
$$

(8)

Note that we assume second-order Markovian property to exploit the context of lower layers. $q_x^{(l)}$ can be derived by a classification network with 256 dimensions, each indicating the probability of an occupancy code. However, $q_f^{(l)}$ is unable to be derived in the same way, because different from $x^{(l)}$ in one-hot representation, $f^{(l)}$ is a numerical vector whose distribution cannot be explicitly represented by a classification network. Therefore, we first apply residual coding to remove the redundancy in $f^{(l)}$:

$$
f^{(l)} = f^{(l)} + r^{(l)}, \quad \text{with} \quad r^{(l)} = g_d(x^{(l-1)}, f^{(l-1)}, x^{(l-2)}, f^{(l-2)}; \psi_f),
$$

(9)

where $g_d$ is a neural network with $\psi_f$ as its parameters, $\tilde{f}^{(l)}$ is the predictable component given its context, which requires no extra bits; and $f^{(l)}$ is the residual. As we have no prior beliefs about $r^{(l)}$, we assume it to be independent and identically distributed (i.i.d.), and model it with a fully-factorized entropy model [14]:

$$
q_f^{(l)} = q(r^{(l)}) = \prod_i^N q(r_i^{(l)}).
$$

(10)

IV. NETWORK ARCHITECTURE

The overall architecture is shown in Figure 3. The proposed network follows a 5-layer architecture, where each layer consists of 5 layers. For each layer, $x^{(l)}$ is the $l$-th octree layer and $e^{(l)}$ is the corresponding occupancy embedding. $f^{(l)}$ is the latent variable of the $l$-th layer. $\tilde{f}^{(l)}$ is the reconstruction residual. $f^{(l)}$ and $\tilde{f}^{(l)}$ respectively are the predicted and reconstructed latent variable. – and + denote the soft subtraction and soft addition operator. $Q$ denotes quantization.
A. Occupancy Embedding

The architecture of the occupancy embedding module is shown in Figure 4(e). The occupancy embedding module takes the one-hot octree layer $x^{(l)}$ as input and outputs the corresponding occupancy embedding $e^{(l)}$. The network aims to encode the semantic information of $x^{(l)}$ into a densely distributed representation that can be better processed by a neural network. As shown in Figure 4(e), the network consists of three sparse convolution layers, which are trained jointly with the entire network. Ablation studies in chapter VI-F demonstrate the effectiveness of the occupancy embedding network, where we can observe that occupancy codes geometrically closer are more similar in the vector space.

B. Encoder Network

The architecture of the encoder network is shown in Figure 4(a). Equation (8) shows that the probability distributions of $x^{(l)}$ and $f^{(l+1)}$ is correlated with latent variable $f^{(l)}$ of the $l$-th layer. Therefore, the encoder network (Figure 4(a)) is designed as a deterministic transform on $x^{(l)}$ and $f^{(l+1)}$ to encapsulate their spatial correlation into $f^{(l)}$:

$$f^{(l)} = g_e(x^{(l)}, f^{(l+1)}, \psi_e),$$

where $\psi_e$ is the encoder parameters. Inspired by Wang et al. [43], we adopt sparse-CNN to design $g_e(\cdot)$ for high-efficiency octree feature extraction. The octree sequence is modeled as a sparse tensor [41], [42]:

$$C = \begin{bmatrix}
    b_1 & x_1 & y_1 & z_1 \\
    b_2 & x_2 & y_2 & z_2 \\
    \vdots & \vdots & \vdots & \vdots \\
    b_N & x_N & y_N & z_N
\end{bmatrix}, \quad F = \begin{bmatrix}
    f_1 \\
    f_2 \\
    \vdots \\
    f_N
\end{bmatrix},$$

where the coordinate matrix $C$ corresponds to the coordinates of the octree nodes, and the feature matrix $F$ corresponds to the occupancy code. The down-scaling of the octree layers is achieved by a stride-two and kernel-size-two sparse convolution layer. Specifically, $f^{(l+1)}$ is first down-sampled by a stride-two sparse convolution layer, followed by several Inception Residual Network (IRN) blocks [49] (Figure 5) for neighborhood feature extraction and aggregation. For the feature fusion of $f^{(l+1)}$ and $x^{(l)}$, the down-sampled $f^{(l+1)}$ is concatenated with the occupancy embedding $e^{(l)}$ and passes through a sparse convolution layer to generate the latent variable $f^{(l)}$ for the $l$-th layer.

C. Soft Subtraction/Addition

The quantization of the residual $r^{(l)}$ introduces error to decoding, leading to noise between the latent variable $f^{(l)}$ and its reconstruction $\tilde{f}^{(l)}$. Therefore, instead of directly adding the quantized residual $r^{(l)}$ to the predicted latent variable $\tilde{f}^{(l)}$ shown in equation (9), we propose soft addition (denoted as $\oplus$) to compensate for the information loss during quantization:

$$\tilde{f}^{(l)} = \tilde{f}^{(l)} \oplus f^{(l)} = \tilde{f}^{(l)} + h_{a}(\tilde{f}^{(l)}, f^{(l)}; \psi_{a}),$$

where $h_{a}(\cdot)$ is a transform with $\psi_{a}$ as its parameters. Similarly, the soft subtraction (denoted as $\ominus$) is defined as follows:

$$r^{(l)} = f^{(l)} \ominus \bar{f}^{(l)} = f^{(l)} - h_{s}(\bar{f}^{(l)}, f^{(l)}; \psi_{s}).$$

D. Decoder Network

The design of the decoder network follows equation (8) and (9). When constructing the network, we assume second-order Markovian property for the system stability, as the lower layer context is potentially informative for the entropy coding of the current layer. To demonstrate the necessity of second-order Markov property, we compare the performance of first- and second-order Markov assumptions in chapter VI-B.

Figure 4(b) shows the architecture of the decoder network, which is divided into two branches corresponding to the decoding of $x^{(l)}$ and $f^{(l+1)}$ respectively:

1) The Decoding of $x^{(l)}$: The context of $x^{(l)}$, including $x^{(l-1)}$, $f^{(l)}$, $x^{(l-2)}$ and $f^{(l-1)}$ are concatenated together and pass through a 2-layer sparse CNN. Finally, a 256-dimensional softmax layer is used to generate the probability distribution of $x^{(l)}$ (denoted as $p^{(l)} \in \mathbb{R}^{N \times 256}$). During training, we directly represent the bit rate of entropy coding with the cross entropy between $x^{(l)}$ and $p^{(l)}$, since cross entropy is a tight lower bound achievable using arithmetic coding algorithms. When evaluating the model, an arithmetic coder is used to encode the difference between $p^{(l)}$ and $x^{(l)}$ to ensure that the decoding of $x^{(l)}$ is lossless.

2) The Decoding of $f^{(l+1)}$: The context $x^{(l)}$, $f^{(l)}$, $x^{(l-1)}$ and $f^{(l-1)}$ are concatenated together and up-sampled to generate the predicted latent variable $\tilde{f}^{(l+1)}$. We adopt a stride-two and kernel-size-two sparse transposed convolution layer to achieve the octree up-sampling. Similar to the encoder network, we utilize several IRN modules for local feature analysis.
and aggregation and a stride-one sparse convolution layer to adjust the dimension of output.

Note that the nodes in the same layer share the same context, meaning that the decoding process of each node is independent of other nodes in the same layer. This independence brings superior parallelism to the entire network. Compared with OctAttention, which requires re-running the model multiple times to overcome the continuous varying of context, our proposed model needs only run once during decoding. Therefore, one can observe significant runtime improvement in the subsequent experiments.

**E. Factorized Deep Entropy Model**

We adopt the factorized deep entropy model [14] for entropy coding. Two operations in the entropy encoding system block the back-propagation of gradients: 1) quantizing the input variable \( y \) into \( \hat{y} \), 2) encoding the quantized variable to and from the bitstream. For 1), the quantization operation is replaced by adding a uniformly distributed noise \( \mu \sim \mathcal{U}(-\frac{1}{2}, \frac{1}{2}) \) to ensure differentiability during training.

For 2), thanks to the development of arithmetic coding, we can approximate the bit rate of encoding \( \hat{y} \) without the actual encoding and decoding process. The estimated size of bitstream is the entropy of \( \hat{y} \), which is a tight lower bound achievable by arithmetic coding algorithms:

\[
R_y = -\log q_y(\hat{y})
\]

We follow [14] and model the probability distribution of \( \hat{y} \) with a fully factorized density model:

\[
q_{y|\phi}(\hat{y}|\phi) = \prod_i q_{y_i|\phi_i}(\hat{y}_i) \mathcal{U}(-\frac{1}{2}, \frac{1}{2}),
\]

where \( q_{y_i|\phi_i} \) is the univariate distribution of the \( i \)-th channel and \( \phi_i \) is the corresponding parameters. In general, adding a hyperprior variable \( z \) upon \( y \) can further capture the spatial correlation of \( y \) for more bit rate reduction [15], [50]. However, it is unnecessary for our proposed model due to the i.i.d. assumption of the residual \( r \) in equation (10). More information can be found in Section VI-E that plots the visualization of \( r \). One can observe the strong independence of \( r \) among nodes in the same layer.

**F. Loss Function**

The model is trained in an end-to-end manner with the following loss function:

\[
\mathcal{L} = \alpha \sum_{l=L-k}^{L} R_y^{(l)} + \sum_{l=L-k}^{L} R_x^{(l)},
\]

where \( k \) is the number of layers, \( R_y^{(l)} \) is the bits per output point bpop to encode the quantized residual \( \hat{r}^{(l)} \) (let \( \hat{r}^{(L-k)} = r^{(L-k)} \) for consistency) of the \( l \)-th layer, \( R_x^{(l)} \) is the bpop to entropy encode \( x^{(l)} \). In experiments, we find that the model easily converges to a local optimum where the bit rate of \( \hat{r}^{(l)} \) is extremely low. Therefore, we add a scaling factor \( \alpha \) to encourage the model to output more informative \( \hat{r}^{(l)} \).

is set as 0.5 for the first several epochs and 0.95 after that. \( R_\gamma^{(l)} \) is computed by the factorized entropy model mentioned in chapter IV-E. \( R_x^{(l)} \) is computed by the binary cross entropy (BCE) between the probability \( p^{(l)} \) and the ground truth \( y^{(l)} \):

\[
R_x^{(l)} = -\frac{1}{N_l} \sum_{i=1}^{N_l} \sum_{j=1}^{256} x_{ij}^{(l)} \log_2 p_{ij}^{(l)},
\]

where \( N_l \) is the number of nodes in the \( l \)-th layer, and \( N_l \) is the number of points in the output point cloud.

**V. EXPERIMENTS**

**A. Datasets**

- **SemanticKITTI** [27] is a public LiDAR dataset for autonomous driving, containing 22 sequences with 43352 scans collected from a Velodyne HDL-64 sensor. We adopt the official data splitting that takes sequences 00 to 10 as the training set and 11 to 21 as the test set.
- **Ford** is a LiDAR dataset suggested by MPEG Common Test Conditions (CTC) [28]. The dataset contains three sequences with 4500 frames. The precision of the original point cloud is 18 bits. We divide sequence 01 as the training set and sequences 02 and 03 as the test set.

**B. Training Strategy**

- For SemanticKITTI, we train four models for octree depth \( L = 12, 11, 10, 9 \). We utilize an Adam optimizer with \( \beta = (0.9, 0.999) \) and the initial learning rate set as 0.0006. We further utilize a learning rate scheduler with a decay rate of 0.7 for every 20 epochs. We train the

![Fig. 6. The visualization of datasets. We randomly choose six scans from two datasets. The upper row is the scans from SemanticKITTI. The lower row is the scans from Ford.](image-url)

| SemanticKITTI | Ford |
|---------------|------|
| -28.27        | -25.55 |
| -25.33        | -22.01 |
| -14.37        | -11.68 |
| -3.95         | -2.98  |

**TABLE I**

| DIFFERENT BD-RATE(%) GAINS AGAINST G-PCC |
|------------------------------------------|
| Ours OctAttention Voxelcontext-Net OctSqueeze | |
| SemanticKITTI | -28.27 | -25.33 | -14.37 | -3.95 |
| Ford | -25.55 | -22.01 | -11.68 | -2.98 |

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
proposed model for 200 epochs, with 1200 iterations for each epoch. For the first 10 epochs, $\alpha$ in equation (17) is set as 0.5. The batch size is set as 4 during training. We conduct all the experiments on a GeForce RTX 3090 GPU with 24GB memory.

- For Ford, we also train four models for octree depth $L = 12, 11, 10, 9$. As Ford is a small dataset with only 1500 frames for training, we first train the $L = 12$ model for 40 epochs. $\alpha$ in equation (17) is set as 0.5 for the first two epochs and 0.95 for the remaining. For the models of $L = 11, 10, 9$, since the amount of quantized data is too small to support the end-to-end training, we perform four epochs of finetuning on the $L = 12$ model on Ford dataset with the initial finetuning learning rate 0.0002. $\alpha$ is set as 0.95 during finetuning.

### C. Baseline Setup

We compare our proposed model with G-PCC and other learning-based LPC compression network including OctAttention, OctSqueeze, VoxelContext-Net and state-of-the-art method OctAttention. Besides, we also compare our method with Feng’s algorithm based on range images.

- For G-PCC, we follow the MPEG CTC to generate the results on SemanticKITTI and Ford with the latest TMC13-v14.
• For OctSqueeze, as the source code is not publicly available, we re-implement their model and train for 200 epochs on SemanticKITTI, 40 epochs on Ford 01.
• For VoxelContext-Net, as the source code is also unavailable, we adopt the re-implemented result by Muhammad et al. [51], which reports better results than [22]. We remove the coordinate refinement module (CRM) in VoxelContext-Net for the fairness of comparison. It is a post-processing module that enhances the quantized point cloud to reduce distortion but is irrelevant from the entropy coding of point cloud geometry.
• For OctAttention, we adopt the result on SemanticKITTI in the paper. Since they have no available results on the Ford dataset, we retrain their model on Ford 01 for 8 epochs (the same number of epochs as SemanticKITTI) to generate the result on Ford 02 and 03.

D. Evaluation Metrics

The bit rate is evaluated using bits per input point (bpip), and the distortion is evaluated using point-to-point geometry (D1) Peak Signal-to-Noise Ratio (PSNR) and point-to-plane geometry (D2) PSNR, which are computed by the software pc_error developed by MPEG. For Ford dataset, we follow the MPEG CTC that sets the PSNR peak value $p = 30000$.

For the floating-point SemanticKITTI dataset, we follow the evaluation metric in [23] that scales the point clouds into...
Fig. 12. Rate-distortion curves on SemanticKITTI and Ford with Point-SSIM measuring distortion. From left to right: SemanticKITTI and Ford.

$[-1, 1]^3$, and sets the PSNR peak value $p = 1$. We also compare the chamfer distance (CD) among all methods:

$$\text{CD}(\mathbf{P}, \tilde{\mathbf{P}}) = \max(\text{CD}'(\mathbf{P}, \tilde{\mathbf{P}}), \text{CD}'(\tilde{\mathbf{P}}, \mathbf{P})),$$

$$\text{CD}'(\mathbf{P}, \tilde{\mathbf{P}}) = \frac{1}{|\mathbf{P}|} \sum_{i} \min_{j} \| \mathbf{p}_i - \tilde{\mathbf{p}}_j \|_2. \quad (19)$$

E. Experiment Results

1) Results on SemanticKITTI: The rate-distortion (D1-PSNR, D2-PSNR, chamfer distance) curves of different methods on SemanticKITTI are shown in Figure 7. We can observe that the four learning-based methods outperform G-PCC due to the optimization by traversing the whole dataset. Besides, OctSqueeze and Voxelcontext-Net assume different degrees of node independence considering the time complexity; thus, neither fully explores the sibling correlation, leading to suboptimal performance. Our method dramatically surpasses the two methods due to the hierarchical latent variable, which encapsulates the correlation of sibling nodes to remove spatial redundancy. Meanwhile, our model surpasses OctAttention, which considers sibling dependence despite the excessive complexity. We also report the corresponding BD-Rate gain of the four learning-based methods against G-PCC in Table I. The experiment result shows that our model achieves state-of-the-art coding efficiency on SemanticKITTI, with the highest $-28.27\%$ BD-Rate gain against G-PCC.

2) Results on Ford: The rate-distortion curves of different methods on Ford are also shown in Figure 7, and the corresponding BD-rate gain against G-PCC is shown in Table I. Ford is a dataset suggested by MPEG CTC. Our proposed model also achieves state-of-the-art performance on Ford, with the highest $-25.55\%$ BD-Rate gain against G-PCC. The experiment result demonstrates the extensibility of our model.

3) Complexity Comparison: In this section, we evaluate the complexity. Figure 8 and Table II compares the decoding time of each method. We can observe that:

- The complexity of OctAttention is extraordinarily high compared to other methods. This is due to the sibling context selection strategy, which causes the context to change during the decoding process and requires multiple runs of the model to update the context. OctAttention requires more than 5 minutes to decode one point cloud frame, which is unaffordable in practical applications.

- On the contrary, our model is end-to-end and fully-factorized, where the nodes in the same layer are independently decoded. Compared with OctAttention, our model is highly parallelized and requires running only once, thus saving more than 99.8\% of decoding time.

- Due to the complexer network structure, our model is slightly slower than OctSqueeze, which assumes the independence of sibling nodes and also decodes layerwisely. But our model explores the correlation of sibling nodes, thus our model achieves better performance.

VI. ANALYSIS AND ABALATION STUDY

A. Ablation Study on Soft Addition/Subtraction

In chapter IV-C, we propose substituting the ordinary addition/subtraction operator for the soft addition/subtraction. Therefore, we perform an ablation study on its effectiveness in Figure 9. We separately train the two models on the training split of SemanticKITTI and evaluate the result on the test split. The experiment result proves that the soft addition/subtraction enables the network to dynamically balance the reconstruction loss and the corresponding bit rate of $\hat{\mathbf{f}}^{(i)}$, thus achieving 3.47\% BD-Rate gain against the ordinary addition/subtraction.

B. Ablation Study on the Markovian Property

In equation (8), we assume the Markovian property of $\mathbf{f}^{(i)}$ and $\mathbf{x}^{(i)}$ for simplification. When establishing the network, we instead assume second-order Markovian property for system stability in the hope that the lower layer context is also informative for the entropy coding of the current layer. We separately train two models of first-order and second-order Markovian properties on the training split of SemanticKITTI and evaluate the result on the test split. In Figure 10, we show the performance comparison of the model based on first-order and second-order Markovian properties. The experimental result proves that the assumption of second-order Markovian property is effective, which achieves 1.09\% BD-Rate gain against first-order Markovian property.

C. Comparison With Range-Image-Based Methods

This chapter compares our network with Feng’s [32] algorithm based on range images. The result is shown in Figure 11. Feng’s algorithm has inferior D1-PSNR and Chamfer Distance compared with Octree-based methods. Compared with D1-PSNR, Feng’s algorithm performs better in terms of D2-PSNR, although the performance is still worse than Octree-based methods. Therefore, our model achieves better performance over Feng’s range-image-based method.

D. Quality Assessment

Various works have emerged in the field of Point Cloud Quality Assessment (PCQA) [52], [53], [54], [55]. This chapter chooses Point-SSIM [53] that measures geometry distortion to evaluate the compression distortion. The result in Figure 12 shows that the visual quality of the highest rate point achieves an average of 0.75 Point-SSIM in SemanticKITTI and 0.78 Point-SSIM in Ford, demonstrating the high quality of reconstructed PCs.
E. Analysis on Latent Variable

Defined by equation (9), the latent variable \( \hat{f}^{(l)} \) of the \( l \)-th octree layer is decomposed into two parts: the predictable component \( \bar{f}^{(l)} \) given the previously decoded context, and the independent residual component \( \hat{r}^{(l)} \) with no prior knowledge. In this section, we visualize the decomposition process of \( \hat{f}^{(l)} \) in Figure 13. The middle-left column is the visualization of \( \hat{f}^{(l)} \), with clearly visible spatial dependence among each node. The middle-right column is the visualization of the residual \( \hat{r}^{(l)} \) that captures the unpredictable component of \( \hat{f}^{(l)} \). We can observe that \( \hat{r}^{(l)} \) consists of a large number of noisy points, which demonstrates the i.i.d. nature of \( \hat{f}^{(l)} \) in equation (10). The rightmost column is the visualization of predictable component \( \bar{f}^{(l)} \) that captures the sibling dependence of \( \hat{f}^{(l)} \), thus we can also observe strong spatial correlation as that in \( \hat{f}^{(l)} \). The visualization indicates that the residual coding successfully captures the dependence of high-scale data into a low-scale variable, and guarantees the i.i.d. nature of the remaining information.

Figure 14 shows the bit rate consumption to encode the latent variable and occupancy, and the effect of the latent variable on encoding occupancy. The bit rate of the latent variable increases slowly with the overall bit rate, accounting for less than 10%. However, the benefit of introducing latent variables as side information is significant. The bit rate of
the feature space, demonstrating that the occupancy embedding module learns the geometric semantics of the occupancy codes.

VII. Conclusion

This paper proposes an end-to-end, fully-factorized LiDAR point cloud compression framework based on the hierarchical decomposition of the octree entropy model. The network utilizes a latent variable to encapsulate the sibling and ancestor dependence, through which the sibling nodes are conditionally independent to be decoded in parallel. We further propose a residual coding framework for the entropy coding of the latent variable. The latent variable is hierarchically downsampled to capture the deeper spatial correlation, and the residual is modeled by a factorized entropy model. We propose to use soft addition/subtraction to substitute ordinary addition/subtraction for the flexibility of the network. The experiment result shows that our framework achieves state-of-the-art performance among all the LPC compression frameworks. It is noted that our framework also shows superior time complexity, saving more than 99.8% decoding time compared with the previous state-of-the-art framework.

REFERENCES

[1] Y. Xu, K. Zhang, L. He, Z. Jiang, and W. Zhu, “Introduction to point cloud compression,” ZTE Commun., vol. 16, no. 3, p. 8. 2018.

[2] W. Zhu, Z. Ma, Y. Xu, L. Li, and Z. Li, “View-dependent dynamic point cloud compression,” IEEE Trans. Circuits Syst. Video Technol., vol. 31, no. 2, pp. 765–781, Feb. 2021.

[3] W. Zhu, Y. Xu, D. Ding, Z. Ma, and M. Nilsson, “Lossy point cloud geometry compression via region-wise processing,” IEEE Trans. Circuits Syst. Video Technol., vol. 31, no. 12, pp. 4575–4589, Dec. 2021.

[4] A. Akhtar, B. Kathariya, and Z. Li, “Low latency scalable point cloud communication,” in Proc. IEEE Int. Conf. Image Process. (ICIP), Sep. 2019, pp. 2369–2373.

[5] G. Zhang, Q. Ma, L. Jiao, F. Liu, and Q. Sun, “AttAN: Attention adversarial networks for 3D point cloud semantic segmentation,” in Proc. 29th Int. Joint Conf. Artif. Intell., Jul. 2020, pp. 789–796.

[6] Y. Yu, W. Zhang, G. Li, and F. Yang, “A regularized projection-based geometry compression scheme for LiDAR point cloud,” IEEE Trans. Circuits Syst. Video Technol., vol. 33, no. 3, pp. 1427–1437, Mar. 2023.

[7] L. Li, Z. Li, S. Liu, and H. Li, “Frame-level rate control for geometry-based LiDAR point cloud compression,” IEEE Trans. Multimedia, early access, Apr. 19, 2022, doi: 10.1109/TMM.2022.3167810.

[8] H. Houshiar and A. Nuchter, “3D point cloud compression using conventional image compression for efficient data transmission,” in Proc. XXV Int. Conf. Inf., Commun. Autom. Technol. (ICAT), Oct. 2015, pp. 1–8.

[9] J. L. Bentley, “Multidimensional binary search trees used for associative searching,” Commun. ACM, vol. 18, no. 9, pp. 509–517, Sep. 1975.

Fig. 15. Analysis on occupancy embedding module. The left column is the occupancy code of octree nodes. The middle column is the visualization of the octree node. The right column is the Euclidean distance of the feature of octree nodes from the octree node 11110000 (240) in feature space.

encoding the occupancy is reduced by more than 40% due to the latent variable. Note that the network autonomously balances the bit rate of each part via end-to-end training, with the final bit rate as the optimization goal.

F. Analysis on Occupancy Embedding Module

To validate the effectiveness of the occupancy embedding module, we compute the feature embedding of different occupancy codes. In Figure 15, we compare the Euclidean distance of three occupancy codes from the base occupancy code 11110000 (240). Note that the input occupancy codes are one-hot vectors whose distance is one between each other. However, after being converted by the embedding module, occupancy codes that are geometrically closer are similar in

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
Yiling Xu (Member, IEEE) received the B.S., M.S., and Ph.D. degrees from the University of Electronic Science and Technology of China, in 1999, 2001, and 2004, respectively. From 2004 to 2013, she was a Senior Engineer with the Multimedia Communication Research Institute, Samsung Electronics Inc., South Korea. She joined Shanghai Jiaotong University (SJTU) and she is now a Professor in the areas of multimedia communication, 3D point cloud compression and assessment, system design and network optimization. She is an Associate Editor of the IEEE TRANSACTIONS ON BROADCASTING. She is also an active member in standards organizations including MPEG, 3GPP, AVS.

Dong Wang (Member, IEEE) received the M.S.E. degree in instrument science and engineering from Southeast University, Jiangsu, China, in 2000. He is now working for Nebula Laboratory of OPPO. His current research focuses on audiovisual compression and processing, including conventional video coding, neural-network based video coding and immersive video coding aspects.

Zhu Li (Senior Member, IEEE) received the Ph.D. degree in electrical and computer engineering from Northwestern University in 2004. He is a Professor with the Department of Computer Science and Electrical Engineering, University of Missouri, Kansas City, and the Director of NSF I/UCRC Center for Big Learning (CBL) at UMKC. He was an AFRL summer faculty with the UAV Research Center, US Air Force Academy (USAFA), from 2016 to 2018, and from 2020 to 2023. He was Senior Staff Researcher with Samsung Research America’s Multimedia Standards Research Laboratory in Richardson, TX, from 2012 to 2015, Senior Staff Researcher with Future Wei Technology’s Media Laboratory in Bridgewater, NJ, from 2010-2012, Assistant Professor with the Department of Computing, the Hong Kong Polytechnic University from 2008 to 2010, and a Principal Staff Research Engineer with the Multimedia Research Laboratory (MRL), Motorola Labs, from 2000 to 2008. He has over 50 issued or pending patents and over 190 publications in book chapters, journals, and conferences in these areas. His research interests include point cloud and light field compression, graph signal processing and deep learning in the next gen visual compression, image processing and understanding. He is an Associate Editor-in-Chief for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEM FOR VIDEO TECH, Associate Editor for IEEE TRANSACTIONS ON IMAGE PROCESSING (2020), IEEE TRANSACTIONS ON MULTIMEDIA (2015 to 2018), and IEEE TRANSACTIONS ON CIRCUITS AND SYSTEM FOR VIDEO TECHNOLOGY (2016-2019). He received the Best Paper Award at the IEEE International Conference on Multimedia and Expo (ICME), Toronto, in 2006, and the IEEE International Conference on Image Processing (ICIP), San Antonio, in 2007.