DeepRAHT: Learning Predictive RAHT for Point Cloud Attribute Compression

Chunyang Fu[1], **Tai Qin**[2], **Shiqi Wang**[1*], **Zhu Li**[3]

[1]City University of Hong Kong, Hong Kong SAR, China
[2]Peking University, Beijing, China
[3]University of Missouri-Kansas City, Kansas City, USA
chunyang.fu@my.cityu.edu.hk, qintai@stu.pku.edu.cn, shiqwang@cityu.edu.hk, zhu.li@ieee.org

Abstract

Regional Adaptive Hierarchical Transform (RAHT) is an effective point cloud attribute compression (PCAC) method. However, its application in deep learning lacks research. In this paper, we propose an end-to-end RAHT framework for lossy PCAC based on the sparse tensor, called DeepRAHT. The RAHT transform is performed within the learning reconstruction process, without requiring manual RAHT for preprocessing. We also introduce the predictive RAHT to reduce bitrates and design a learning-based prediction model to enhance performance. Moreover, we devise a bitrate proxy that applies run-length coding to entropy model, achieving seamless variable-rate coding and improving robustness. DeepRAHT is a reversible and distortion-controllable framework, ensuring its lower bound performance and offering significant application potential. The experiments demonstrate that DeepRAHT is a high-performance, faster, and more robust solution than the baseline methods.

Project Page — https://github.com/zb12138/DeepRAHT

Introduction

A 3D point cloud is a set of points that consists of geometry (*i.e.*, position of points) and attributes (*e.g.*, color, normal vectors, and reflectance). As a common representation of 3D data, point clouds have been widely applied in various practical applications such as mixed reality, autonomous vehicles, and high-resolution mapping (MPEG 3DG 2016). However, the massive points' unstructured and unordered nature presents challenges for existing compression techniques. The point cloud geometry and attribute compression can be performed jointly (Alexiou, Tung, and Ebrahimi 2020; Yu et al. 2025), but the dominant approaches in recent research (Wang et al. 2025a,b; Guo et al. 2024; Fang et al. 2022) and the MPEG geometry-based point cloud compression standard G-PCC (Schwarz et al. 2018) compress the geometry first and then compress the attributes based on the reconstructed geometry. The geometry compression of point clouds has made monumental progress in recent research (Fu et al. 2022; Song et al. 2023b; Wang et al. 2025a; Wang and Gao 2025; You et al. 2025), particularly in the various deep learning-based approaches. However, the attribute compression has not been sufficiently explored. This paper studies the lossy compression of point cloud attributes, assuming that the geometry has been reconstructed.

Point cloud attributes are essential and have various applications. For instance, colors provide the most basic visual information, reflectance is used for object detection in LiDAR, and normals are utilized for surface reconstruction and rendering. Furthermore, the recently popular Gaussian splatting data (Kerbl et al. 2023) is also related to point clouds, as the parameters of the Gaussian primitives can be viewed as the attributes. One important method for lossy point cloud attribute compression is the Region-Adaptive Hierarchical Transform (RAHT) (De Queiroz and Chou 2016), which has been adopted as the core method in the G-PCC standard (MPEG 3DG 2023), demonstrating excellent complexity and performance. However, only a few research about deep learning studies on RAHT. 3DAC (Fang et al. 2022) is the first framework that uses a deep entropy model for coding RAHT transform coefficients, but it merely uses the manually crafted RAHT (*i.e.*, same as the original implementation) to generate transform coefficients first. Then a deep entropy model was devised to learn the distribution of coefficients. Since the manual RAHT is non-differentiable, 3DAC can only optimize the bitrate. 3DAC also ignored the predictive RAHT, a crucial improvement in the current G-PCC RAHT standard (MPEG 3DG 2023). Prediction can significantly reduce the uncertainty of transform coefficients, and thus, encoding the residuals, not the coefficients, can remarkably reduce the bitrate. However, the learning of the prediction of RAHT has been largely overlooked. The lack of end-to-end framework limits the application of RAHT in deep learning, such as joint learning with RAHT and other components (*e.g.*, deblocking task).

In this paper, we propose the first end-to-end differentiable RAHT framework called **DeepRAHT**. We also integrated a learnable prediction model. In our method, the multi-scale RAHT transform is performed during the learning reconstruction process, without requiring a manual RAHT in advance. The end-to-end design provides high application potential for our framework, allowing joint optimization of RAHT and other components. DeepRAHT has an identical framework to the MPEG G-PCC and builds the

*corresponding author

performance lower bound on it. Furthermore, DeepRAHT is entirely reversible and distortion controllable. Our main contributions are as follows:

- We implement the first end-to-end differentiable RAHT framework, which has an identical structure to the G-PCC reference software (*i.e.*, tmc13v14, *aka* G-PCCv14) and establishes a performance lower bound on it.
- We develop a learnable prediction model incorporating grandfather scale context. This model validates the learnability of DeepRAHT and achieves over 24% BD-rate gains compared to G-PCCv14.
- We propose a rate proxy to utilize run-length coding as the entropy coder, replacing the entropy bottleneck (Ballé et al. 2018) used in most works. Run-length coding is more robust for our framework and achieves seamless variable-rate coding.

Related Work

Traditional PCAC Methods

The key aspect of point cloud attribute compression (PCAC) is to explore attribute correlations through geometry structures. Mainstream PCAC methods primarily utilize prediction, projection, and transformation techniques. The prediction tree (Waschbüsch et al. 2004) is the earliest related method in prediction-based methods. G-PCC (MPEG 3DG 2023) predicting transform branch also performs predictions in refinement levels divided by levels of detail. Projection-based methods, such as MPEG V-PCC (Graziosi et al. 2020), convert 3D point sets into 2D images, enabling the application of existing image and video compression standards (e.g., JPEG and H.265) (Mekuria, Blom, and Cesar 2016; Li et al. 2020). Transformation-based methods focus on compressing attributes in the frequency domain, where energy is more concentrated. Graph Fourier Transform (GFT) and its variants (Cohen, Tian, and Vetro 2016; Shao et al. 2017; Xu et al. 2020; Song et al. 2023a; Zhang et al. 2024) have proven effective but are too complex when performing eigen decomposition. Queiroz *et al.*, (De Queiroz and Chou 2016) propose an efficient adaptive Haar wavelet transform called Region-adaptive Hierarchical Transformation (RAHT), which avoids the eigen decomposition in GFTs and is adopted in the MPEG G-PCC standard. Our framework achieves the parallelization of RAHT, resulting in the low complexity of DeepRAHT.

Learning-based PCAC Methods

Alexiou *et al.* (Alexiou, Tung, and Ebrahimi 2020) pioneered compressing point cloud attributes using dense 3D convolution. Following this, Deep-PCAC(Sheng et al. 2022) introduced another approach based on point convolution. FoldingNet (Quach, Valenzise, and Dufaux 2020) projected point clouds into 2D images, applying image codecs for compression. LVAC (Isik et al. 2022) utilized Implicit Neural Representation (INR) for point cloud attribute compression. Fang *et al.* (Fang et al. 2022) proposed 3DAC, the first learning-based method to encode RAHT coefficients directly; however, the RAHT used was non-differentiable and lacked the prediction component. Concurrently, SparsePCAC(Wang and Ma 2022) made progress by leveraging stacked sparse convolutional layers. Despite these innovations, their lossy compression performance still lagged the early G-PCC test model (*i.e.*, G-PCCv14).

Recently, some methods have made significant progress and claim to outperform G-PCCv14. 3CAC (Wang, Ding, and Ma 2023) and CNet (Nguyen and Kaup 2023) were the early attempts for lossless PCAC. TSC-PCAC (Guo et al. 2024) enhanced SparsePCAC by incorporating Transformer-based modules for inter-channel context regression. Li *et al.* (Li et al. 2024) introduced the first network based on graph dictionary learning. Mao *et al.* proposed SPAC(Mao et al. 2025) for lossy PCAC, a sampling-based framework utilizing residual networks and multiple slices. Recently, Unicorn (Wang et al. 2025b) stands as the state-of-the-art deep learning framework for point cloud compression, employing average pooling to obtain multiple scales and sparse convolutions to code attribute residuals. These methods demonstrated superior results across most datasets, although specific cases revealed some robust issues.

Methodology

Overview

Given a 3D voxelized point cloud with N points denoted by $P = (p, a)$, $p \in \mathbb{N}^{N\times 3}$ is the set of points' positions and $a \in \mathbb{R}^N$ is the corresponding attribute[1]. Suppose the input point cloud is P_0, we aim to lossily compress a_0 with known p_0. In DeepRAHT, P_0 is first pooled s times by sum-pooling with stride $2 \times 2 \times 2$, and thus generates s scales $\{P_1, ..., P_m, ..., P_s\}$. Define A_m as the sum of attributes associated with points encompassed by the node in P_m, thus from the definition of sum-pooling,

$$A_{m+1,i} = \sum_{j\in\mathcal{N}_i} A_{m,j}, \;\; m = 0, 1, ..., s-1, \qquad (1)$$

where i is the parent node in P_{m+1} with at most eight child nodes j in P_m after pooling. $A_0 = a_0$ is the input attributes.

The overall framework of DeepRAHT is illustrated in the Fig. 1. Initially, sum-pooling is performed s times to generate multi-scale point clouds. The encoding process starts from the last scale s in a top-down manner. Point cloud at each scale is applied with the **T**ransform model, while the **P**rediction model is optional (*e.g.*, prediction is disabled in scale m for illustration). In the **T** model, A_m is decomposed into the alternating coefficient AC_m and the direct coefficient DC_m using the **Haar** module. The AC_m or its residual is encoded and decoded by the entropy coder **EM**, while DC_m (excluding the root DC_s) is equivalent to A_{m+1} and has been encoded at a higher scale and thus discarded.

Meanwhile, reconstruction also proceeded while encoding. The **iHaar** module performs an inverse Haar transform to reconstruct $\hat{A}_m$ using the decoded $\widehat{AC}_m$ and the reconstructed $\widehat{DC}_m$, where $\widehat{DC}_m$ is reconstructed from the higher scale $\hat{A}_{m+1}$. The reconstructed $\hat{A}_m$ is subsequently

[1]A single channel of attributes is discussed here, which is more general and the other channels are similar.

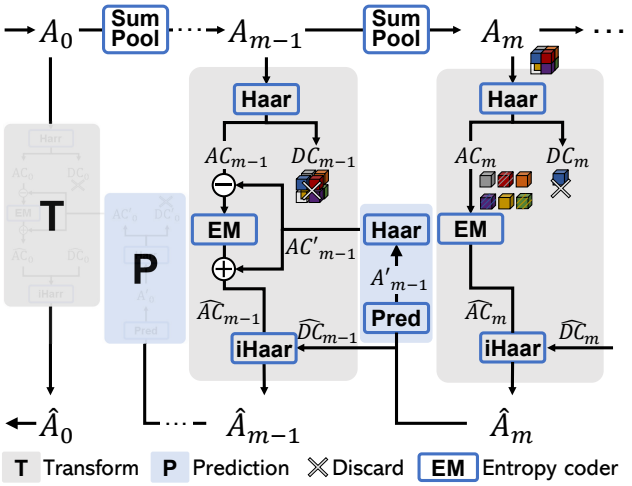


Figure 1: Overview of DeepRAHT. A_0 is the input attributes and $\hat{A}_0$ is the reconstructed attributes.

Notation	Description
A	sum of attributes
w	number of points in node
$a = A/w$	averaged attributes
$g = A/\sqrt{w}$	normalized attributes
$g_{\text{L}}, ..., g_{\text{HHH}}$	transform coefficients

Table 1: Notations.

used to reconstruct $\widehat{DC}_{m-1}$ similarly. In the **P** model, $\hat{A}_m$ also predicts AC'_{m-1} for the residual coding of AC_{m-1}. The processes **T** and **P** are repeated until encoding A_0.

The decoding process also starts from scale s and is consistent with the reconstruction process, and $\widehat{A}_0$ represents the decoded point cloud attributes. Table 1 is the notations.

Transform Model

This section demonstrates the **Haar** and **iHaar** modules and their convolution-based implementation. Region Adaptive Hierarchical Transform (RAHT) is a variation of the Haar wavelet transform, and it was first proposed in (De Queiroz and Chou 2016). In this paper, we use a dyadic version of RAHT (Jonathan Taquet 2020), which improves decorrelation of the high-frequency coefficients in original RAHT.

For each $2 \times 2 \times 2$ voxel, the Haar transform decomposes eight nodes into one direct coefficient and seven alternating coefficients. Fig. 2 shows an example, where nodes are with indexes j at scale m. Note that one of them is transparent, indicating that it encompasses no points. Denote g_m as the normalized attributes as the node features for the input,

$$g_{m,j} = A_{m,j}/\sqrt{w_{m,j}}, \quad j \in \mathcal{N}_i. \tag{2}$$

$w_{m,j}$ is the number of points of p_0 encompassed by node. $A_{m,j}$ is the sum of attributes of those points, and i is the index of output DC (node LLL) in P_{m+1}, defined in Eqn. (1).

Z-axis decomposition is applied first and illustrated as an example. The normalized attributes $g_1, g_2 \in \{g_{m,j}\}$ of the

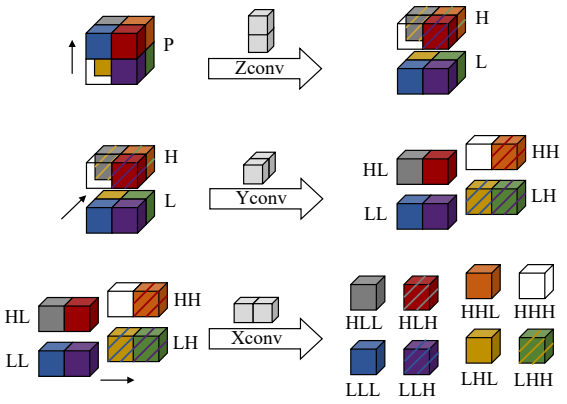


Figure 2: Dyadic RAHT decomposition.

lower and upper node pairs (*e.g.*, purple and red) are transformed according to the following formula,

$$\begin{bmatrix} g_{\text{L}} \\ g_{\text{H}} \end{bmatrix} = \frac{1}{\sqrt{w_1 + w_2}} \begin{bmatrix} \sqrt{w_1} & \sqrt{w_2} \\ -\sqrt{w_2} & \sqrt{w_1} \end{bmatrix} \begin{bmatrix} g_1 \\ g_2 \end{bmatrix}, \tag{3}$$

where weight w_1 is the number of the points encompassed by the lower node (*e.g.*, purple node) and w_2 is the counterpart of the upper node (*e.g.*, red node). g_{L} and g_{H} are the output coefficients for the low frequency and high frequency coefficients (*i.e.*, shaded) nodes, and they will serve as the normalized attributes for the following decompositions. The weights of the output nodes are assigned as $w_{\text{L}} = w_{\text{H}} = w_1 + w_2$. Note that if one of the nodes in the pair is empty, the normalized attributes of the nonempty node will transfer directly to the low frequency, to be forced to undergo further decomposition. The counterpart of high frequency is set to zero: $g_{\text{L}} = g_{\text{nonempty}}, g_{\text{H}} = 0, w_{\text{H}} = 0$. After this decomposition, the input coefficients are converted to half low-frequency and half high-frequency coefficients. Similarly, two Y-axis decompositions for the L, H nodes, and four X-axis decompositions for the LL, LH, HL, HH nodes, are applied sequentially according to Eqn. (3).

After the decompositions of the three axes, g_m are transformed into eight groups of coefficients $g_{\text{LLL}}, g_{\text{LLH}}, ..., g_{\text{HHH}}$. Note that their quantities are not equal due to the empty nodes. g_{LLL} group constitutes the direct coefficient (DC), and it can be proved that DC is equivalent to the normalized attributes of the following scale (see supplementary for the proof),

$$DC_{m,\mathcal{N}_i} \equiv g_{\text{LLL}_{m,\mathcal{N}_i}} = A_{m+1,i}/\sqrt{w_{m+1,i}} = g_{m+1,i}, \tag{4}$$

where $w_{m+1,i} = \sum_{j \in \mathcal{N}_i} w_{m,j}$ and $A_{m+1,i}$ is defined in Eqn. (1). In this regard, in terms of DC, the Haar transform is equivalent to the sum-pooling with a stride of $2 \times 2 \times 2$ described in Eqn. (1), and followed by a square-root normalization. From this equation, DC_m is equivalent to A_{m+1}, thus DC_m is discarded at encoding, because A_{m+1} has been encoded in the higher scale. Specifically, the root DC in the final scale (*i.e.*, DC_s) is encoded directly. The other seven coefficients constitute alternating coefficients (AC) and will be compressed using the entropy coder.

In the reconstruction, from Eqn. (4), DC can be reconstructed from the decoded attributes from the higher scale

$$\widehat{DC}_m = \widehat{A}_{m+1}/\sqrt{w_{m+1}}, \tag{5}$$

and AC are decoded from the bitstream by the entropy model. The **iHaar** module operates in the reverse manner and finally obtains reconstructed attributes $\widehat{A}_m$.

Implementation G-PCC (MPEG 3DG 2023) implements the above RAHT by a manual framework based on C++, and it is not differentiable. We use Minkowski Sparse Tensor (Choy, Gwak, and Savarese 2019) to build the differentiable dyadic RAHT. A sparse tensor $\{C, F\}$ has coordinate field $C \in \mathbb{N}_0^{N\times 3}$ and associated features $F \in \mathbb{R}^{N\times k}$. Thus, the sparse tensor $\boldsymbol{A}_0$[2] for A_0 can be defined as $\{p_0, a_0\}$. Then the formula Eqn. (1) is equivalent to

$$\boldsymbol{A}_{m+1} = \mathrm{SumPooling}(\boldsymbol{A}_m, k = 2^3, s = 2^3), \tag{6}$$

where k means the kernel size and s means the stride. The SumPooling can also be used to calculate the number of points encompassed by nodes. Define the initial weight sparse tensor $\boldsymbol{w}_0 = \{p_0, \mathbf{1}\}$, then

$$\boldsymbol{w}_{m+1} = \mathrm{SumPooling}(\boldsymbol{w}_m, k = 2^3, s = 2^3). \tag{7}$$

To model Haar decompositions, we use sparse convolution to model each axis decomposition. Given Z-axis as an example, the details are shown in the **ZHaar** of Fig. 3, where Zconv is a sparse convolution and it is defined as,

$$\mathrm{Zconv} \equiv \mathrm{Conv}(i = 1, o = 2, k = s = (1, 1, 2)), \tag{8}$$

where i means input channels, o means output channels. The initial weights for the convolution kernel are predefined as the identity matrix $\boldsymbol{I}_2$. In this way, nodes g_1 and g_2 are obtained and mapped to the two output channels. The definitions of Yconv and Xconv are similar, but the kernel size and stride are $(1, 2, 1)$ and $(2, 1, 1)$, respectively.

Denote $\mathrm{f} \in \{\mathrm{P}, \mathrm{L}, \mathrm{H}, \mathrm{LL}, .., \mathrm{HH}, \mathrm{LLL}, ...\mathrm{HHH}\}$ (omit the default scale m) as the frequencies in Fig. 2, and denote w_f as the weight (points number) for each node. $w_\mathrm{L} \ldots, w_\mathrm{HH}$ can be drawn by,

$$\boldsymbol{w}_1, \boldsymbol{w}_2 = \mathrm{HaarConv}(\boldsymbol{w}_\mathrm{f}) \tag{9}$$

$$\boldsymbol{w}_\mathrm{fL} = \boldsymbol{w}_1 + \boldsymbol{w}_2 \tag{10}$$

$$\boldsymbol{w}_\mathrm{fH} = \begin{cases} \mathbf{0} & \text{if } w_1 = 0 \text{ or } w_2 = 0 \\ \boldsymbol{w}_\mathrm{fL} & \text{otherwise,} \end{cases} \tag{11}$$

where $\mathrm{HaarConv} \in \{\mathrm{Zconv}, \mathrm{Yconv}, \mathrm{Xconv}\}$ according to Fig. 2. Note that $\boldsymbol{w}_\mathrm{P} = \boldsymbol{w}_m$ as the initial weights. fL and fH mean corresponding low and high frequency of the base frequency (*e.g.*, if f = P, fH = H; if f = H, fL = HL).

Similarly, Eqn. (9) can be used to calculate the transform coefficients $\boldsymbol{g}_\mathrm{f}$. Therefore, the formula Eqn. (3) for forward Haar can be implemented by,

$$\boldsymbol{g}_1, \boldsymbol{g}_2 = \mathrm{HaarConv}(\boldsymbol{g}_\mathrm{f}) \tag{12}$$

$$\boldsymbol{g}_\mathrm{fL} = \alpha * \boldsymbol{g}_1 + \beta * \boldsymbol{g}_2 \tag{13}$$

$$\boldsymbol{g}_\mathrm{fH} = \begin{cases} \mathbf{0} & \text{if } w_1 = 0 \text{ or } w_2 = 0 \\ -\beta * \boldsymbol{g}_1 + \alpha * \boldsymbol{g}_2 & \text{otherwise,} \end{cases} \tag{14}$$

[2]Bold font represents sparse tensor and italic font denotes its features $A = \boldsymbol{A}.F$.

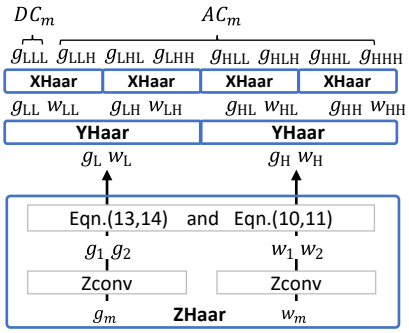


Figure 3: Details of **Haar** module.

where,

$$\alpha = \sqrt{w_1/(w_1 + w_2)}, \beta = \sqrt{w_2/(w_1 + w_2)}. \tag{15}$$

Note that $\boldsymbol{g}_\mathrm{P} = \boldsymbol{g}_m$. The details of **Haar** module is shown in Fig. 3. Eqn. (9)∼(15) are repeated as f from P to HH until $DC_m = g_\mathrm{LLL}$ and $AC_m = \{g_\mathrm{LLH}, ..., g_\mathrm{HHH}\}$ are obtained.

The **iHaar** module employs ConvolutionTranspose to model the inverse of HaarConv, *i.e.*,

$$\mathrm{iZconv} \equiv \mathrm{ConvT}(i = 2, o = 1, k = s = (1, 1, 2)). \tag{16}$$

The implementation for the backward Haar is,

$$\boldsymbol{g}_1 = \alpha * \boldsymbol{g}_\mathrm{fL} - \beta * \boldsymbol{g}_\mathrm{fH} \tag{17}$$

$$\boldsymbol{g}_2 = \beta * \boldsymbol{g}_\mathrm{fL} + \alpha * \boldsymbol{g}_\mathrm{fH} \tag{18}$$

$$\boldsymbol{g}_\mathrm{f} = \mathrm{iHaarConv}([\boldsymbol{g}_1, \boldsymbol{g}_2]), \tag{19}$$

where α, β are defined in Eqn. (15), $\mathrm{iHaarConv} \in \{\mathrm{iXconv}, \mathrm{iYconv}, \mathrm{iZconv}\}$, and f is from HH to P. Finally, attributes sum is reconstructed by $\widehat{A}_m = g_\mathrm{P} * \sqrt{w_m}$.

Prediction Model

In DeepRAHT, we encode from the coarsest layer P_s to the finest layer P_0, and thus we can incorporate the prediction model. G-PCC version 14 introduces predictive RAHT in (Sébastien Lasserre 2019), which up-samples the attributes of the current node by parent neighbors based on the inverse distance weighted prediction. In current G-PCC version 23, Wang *et al.* (Wang et al. 2023) introduce utilizing the decoded sibling neighbors to enhance prediction accuracy. However, this auto-regressive technique will significantly increase decoding time if applied to batch processing-based methods in deep learning. Therefore, we only use the parent scales to design the prediction model as in G-PCCv14. We first introduce a learning implementation of IDW and then propose a prediction compensation module to enhance the prediction performance. Experiment results indicate the learning-based **Pred** model can achieve an even better performance than that of sibling-based prediction.

IDW Prediction We design a sparse convolution for the inverse distance weighted (IDW) prediction. To ensure stability, IDW is performed in the averaged attributes domain. Suppose $\boldsymbol{a}'_{m-1} = \mathrm{IDW}(\widehat{\boldsymbol{a}}_m)$ is the predicted averaged attributes of scale $m - 1$, and it is implemented as,

$$\mathrm{IDW}(\widehat{\boldsymbol{a}}_m) \equiv \mathrm{Conv}(\mathrm{Unpool}(\widehat{\boldsymbol{a}}_m), k = 3^3, s = 1^3) \tag{20}$$

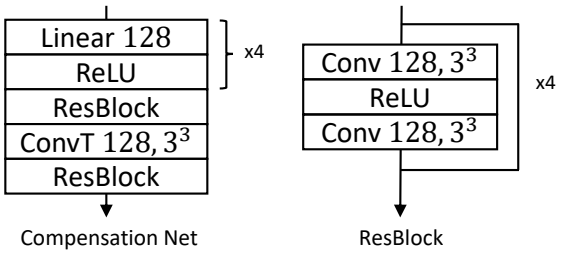


Figure 4: Prediction compensation module.

where $\widehat{\boldsymbol{a}}_m \equiv \widehat{\boldsymbol{A}}_m/w_m$ is the averaged attributes, and $\widehat{\boldsymbol{A}}_m$ is the reconstructed attributes from parent scale, Unpool is an unpooling operation with stride 2. The Conv is a normalized convolution where the weights of the convolution kernel sum to 1 and the proportions are predefined based on the distance from the kernel center. Specifically, the proportions of weights for center, faces, edges, and angles in kernel are set to 4, 3, 2, and 1.

Prediction Compensation As mentioned earlier, we do not utilize the sibling nodes to facilitate prediction. Instead, we propose a compensation module to enhance the prediction by leveraging context from the grandparent (*i.e.*, $m+1$) scale. This approach avoids the auto-regression problem, thereby not increasing the decoding time too much. We use the prediction error between the grandfather and the father scales to compensate for the current prediction. Combining with Eqn. (20), the prediction after compensation is,

$$\boldsymbol{a}'_{m-1} = \mathrm{Comp}(\widehat{\boldsymbol{a}}_m - \mathrm{IDW}(\widehat{\boldsymbol{a}}_{m+1})) + \mathrm{IDW}(\widehat{\boldsymbol{a}}_m), \quad (21)$$

where Comp is the compensation module, which is shown in the Fig. 4 (omit the linear header of input and output). The prediction compensation module is designed empirically and composed of multiple stacked linear layers and convolutions with hidden layer dimension 128, kernel size 3^3, including a transposed convolution with a stride of 2.

The predicted averaged attributes $\boldsymbol{a}'_{m-1}$ yields predicted sum of attributes $\boldsymbol{A}'_{m-1} = \boldsymbol{a}'_{m-1}w_{m-1}$, and Haar transform is also applied to predict the AC coefficients $AC'_{m-1} = \textbf{Haar}(\boldsymbol{A}'_{m-1})$ as shown in the **P** model. Finally, the residuals of AC,

$$r_{m-1} = AC_{m-1} - AC'_{m-1} \quad (22)$$

are encoded.

Note that the prediction compensation can be removed based on the prediction performance, which can be evaluated on mean square error, with a cost of s bits to signal to the decoder. This mechanism improves the robustness of DeepRAHT and ensures that the performance lower bound is G-PCCv14. The removal hardly occurred in the experiments, which indicated that the prediction compensation module was always practical.

Entropy Coder

Most deep learning-based compression methods utilize bottleneck (Ballé et al. 2018) in their entropy models. However, it is greatly affected by the variance of the data. To address this issue, we use the zero run-length coding (MPEG 3DG 2020) (see supplementary for the details) as the entropy coder. Zero run-length coding is efficient for compressing data where there are large numbers of zeros. The coefficient residuals of RAHT are highly concentrated around zeros and thus zero run-length coding performs better, converges faster, and is more robust. Since the run-length coding is not differentiable, we propose a rate proxy based on the Laplace distribution to estimate its probabilistic model,

$$q(r) = \int_{r-0.5}^{r+0.5} \mathcal{L}_{\mu,\sigma}(r)\, dr, \quad (23)$$

where the mean μ and standard variance σ are determined by experiments. The actual bitrate of run-length coding is approximately equal to the cross-entropy $R \approx \alpha H(p,q)$, where p is the distribution of quantized residuals $\mathrm{Q}(r/qs)$, and α represents the proportion by which the run-length coding outperforms the proxy with the assumed Laplace distribution. Q is a $\mathrm{STE\_ROUND}$ (Bengio, Léonard, and Courville 2013), and qs is the quantization step. Note that the Haar transform in DeepRAHT is entirely reversible and thus the distortion comes from the quantization of the residuals, which only depends on the quantization step.

Learning

Since **Haar**, **iHaar**, **Pred** as well as the **EM** modules are end-to-end differentiable, the final reconstruction error can directly be obtained by $\ell_{recon} = \|a_0 - \widehat{a}_0\|_2^2$. In the prediction model, the prediction loss $\ell_{pred} = \sum_m \|(a_m - a'_m)\|_2^2$ can also facilitate convergence. On the other hand, the loss of bitrate can be drawn from the rate proxy $\ell_{bits} = -\sum_m \log_2 q(r_m/qs)$, where $q(r)$ is defined in Eqn. (23). Thus, the total loss is,

$$\ell = \ell_{bits} + \lambda(\ell_{recon} + \ell_{pred}). \quad (24)$$

λ is the weight to balance the distortion and bitrate loss.

Experiments

Experimental Setup

Datasets We train our model on the RWTT (De Queiroz and Chou 2016) dataset suggested in MPEG (MPEG 3DG 2024), which contains 568 real-world objects. The evaluation is performed on the first frames of Owlii (Xu, Lu, and Wen 2017), 8iVSLF (Krivokuća, Chou, and Savill 2018), and MPEG CTC Samples (MPEG 3DG 2022), which are popular testing sets in most of the PCAC works.

Evaluation Metrics The peak signal-to-noise ratio (PSNR) is employed to assess reconstruction quality, and bits per point (bpp) is used to evaluate the compression ratio. The Bjøntegaard Delta Bit Rate (BD-BR) is used to assess the overall bitrate saving under the same quality.

Baselines **G-PCC** (MPEG 3DG 2023) is the most widely used standard provided by MPEG. The latest public version (tmc13v23) on the Octree-RAHT branch is used for comparison. **3DAC** (Fang et al. 2022) is the first learning-based method to code the manual RAHT coefficients. **TSC-PCAC** (Guo et al. 2024) is an auto-encoder method based

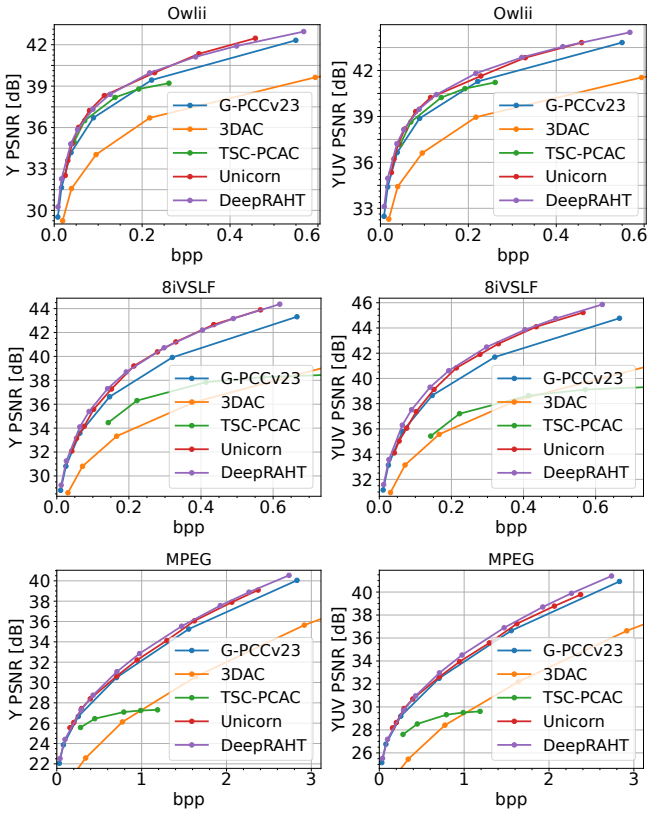


Figure 5: R-D curves averaged over the datasets.

Anchor	G-PCCv23		3DAC		TSC-PCAC		Unicorn	
	Y	YUV	Y	YUV	Y	YUV	Y	YUV
basketball	-23.9	-24.8	-69.1	-68.7	-26.4	-31.0	-7.8	-11.0
dancer	-24.4	-25.5	-72.2	-71.4	-19.2	-26.6	-5.8	-10.9
exercise	-14.3	-16.8	-64.9	-64.5	-0.4	-2.7	-10.0	-13.5
model	-13.0	-13.0	-62.6	-61.9	0.2	9.2	8.0	7.1
Owlii AVG.	-18.9	-20.0	-67.2	-66.6	-11.4	-12.8	-3.9	-7.1
Thaidancer	-13.2	-12.3	-62.9	-63.6	-34.8	-47.4	8.0	3.0
boxer	-23.0	-25.9	-76.4	-75.9	-82.5	-89.8	-20.5	-25.7
longdress	-17.3	-16.5	-67.7	-68.3	-51.8	-61.7	1.1	-6.6
loot	-16.6	-20.0	-73.0	-73.0	-66.0	-73.1	-6.9	-12.0
redandblack	-14.5	-13.3	-76.5	-76.6	-62.6	-72.3	-8.6	-20.3
soldier	-16.1	-17.2	-68.7	-67.7	-46.5	-66.6	-0.8	-3.8
8iVSLF AVG.	-16.8	-17.5	-70.9	-70.9	-57.4	-68.5	-4.6	-10.9
Egyptian	-8.6	-8.1	-43.0	-44.0	-83.8	-91.6	NA	NA
Facade	-16.1	-15.3	-76.9	-78.1	-63.2	-63.6	8.0	7.1
House	-4.1	-3.8	-79.1	-81.7	-74.9	-85.4	-6.9	-6.5
Shiva	-2.3	-1.5	-41.6	-44.0	-50.9	-58.6	-10.4	-10.6
ULB	-15.3	-14.7	NA	NA	-69.6	-76.3	NA	NA
queen	-29.4	-29.6	-64.9	-65.8	-70.7	-79.6	NA	NA
Staue	-9.1	-7.9	-60.8	-62.5	-50.9	-57.1	-4.5	-5.9
MPEG AVG.	-12.1	-11.6	-61.1	-62.7	-66.3	-73.2	-3.4	-4.0
Average	-15.9	-16.4	-66.4	-66.7	-45.0	-51.5	-4.0	-7.3

NA: *The method fails to compress the data.*

Table 2: BD-BR gain (%) against with baseline methods.

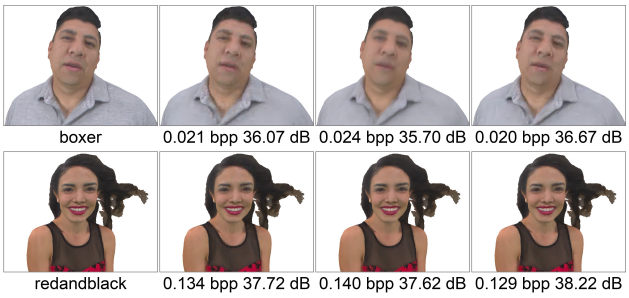
on the Transformers and inter-channel context regression. **Unicorn** (Wang et al. 2025b) is currently the state-of-the-art deep learning-based point cloud compression framework. All the learning-based methods are trained on RWTT.

Implementation Details The transform and prediction are implemented by Pytorch, and the run-length coding is written by C++. Point cloud colors are compressed in YUV space. The training and testing of all methods are performed with a Core i9-13th CPU and one NVIDIA 4090 GPU (24 GB memory). The total scale s is set to the geometry precision of the point cloud, and prediction starts from $s-2$ to 0. We set $qs = 8, \lambda = 1/255$, batch sizes to 1, and use Adam optimizer with a learning rate of 0.0001 for training. The validation losses details are in the supplementary.

Experiment Results

Quantitative BD-BR Gains The quantitative comparison of DeepRAHT with baseline methods is presented in Table 2. DeepRAHT outperforms all baseline methods, achieving a 16.4% improvement over G-PCCv23 and a 7.3% YUV BD-BR improvement over Unicorn on average. The averaged R-D curves are shown in Fig. 5. DeepRAHT outperforms all the baseline methods across a wide bitrate range. Meanwhile, DeepRAHT demonstrated a significant improvement in chrominance, achieving a BD-BR gain of 20.5% on U and 20.8% on V compared to Unicorn. Detailed R-D curves are provided in the supplementary. Notably, DeepRAHT is more robust and successfully compressed and surpasses G-PCC across all data, while other learning-based methods struggled with certain large or sparse point clouds.

Qualitative Visualization The visualization comparison is shown in Fig. 6. It can be observed that DeepRAHT significantly reduces artifacts compared to G-PCC. Unicorn produces smoother reconstructions but irreversibly loses details, such as the texture of the boxer's shirt and facial details of the redandblack, while DeepRAHT preserves more details. This advantage comes from the fact that DeepRAHT is reversible, where the distortion only depends on the quantization. More comparisons are provided in the supplementary.



(a) GT (b) G-PCC (c) Unicorn (d) DeepRAHT

Figure 6: Ground Truth (GT) and reconstructions of G-PCC, Unicorn and DeepRAHT. Bpp and Y PSNR are provided.

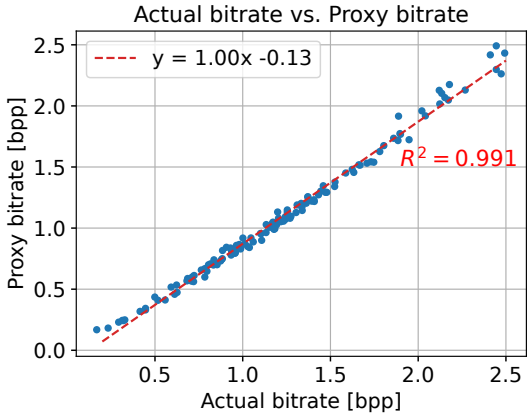


Figure 7: Bitrate of run-length coding and the proxy.

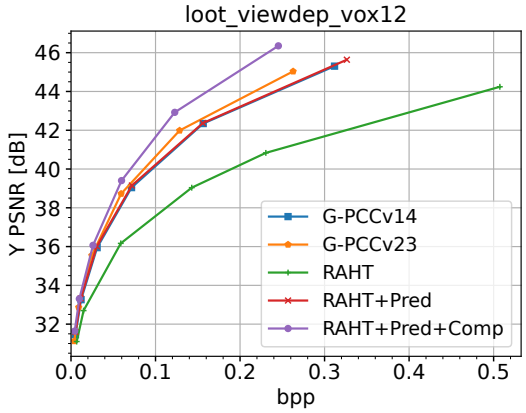


Figure 8: Ablation study on loot_viewdep.

Rate Proxy of Run-length Coding We propose a rate proxy within the entropy model to predict the rate of run-length coding. We obtain the parameters $\alpha = 0.425, \mu = 0, \sigma = 0.2$ in Eqn. (23) by fitting the actual data. Fig. 7 from the testing data illustrates the actual bitrate alongside the predicted values. The coefficient of determination is 0.991, demonstrating the accuracy of the proposed rate proxy.

Variable-rate Coding Variable-rate coding requires a single neural model to achieve lossy compression at varying rates and qualities. In contrast, 3DAC and TSC require training multiple models for variable rates because the entropy model they use heavily depends on the variance of the data. Unicorn aims to address this challenge by introducing the Adjustable Quantization Layer; however, a single model can only accommodate approximately three bitrates. We propose run-length coding to resolve this issue, demonstrating strong robustness to data with varying variance under a Laplace distribution. As a result, the rate points in Fig. 5 are obtained by adjusting $qs = \{8, 10, 12, 16, 24, 32, 48, 64, 128, 224\}$ on one highest rate checkpoint without needing extra training.

Complexity The complexity comparison was evaluated and averaged across all bitrates and frames on the 8iVSLF dataset, which contains an average of 3251505 points per point cloud. The test point clouds are divided into blocks within $2 * 10^6$ points by KD-tree to avoid memory overflow. As shown in Table 3, DeepRAHT is faster than all the baseline methods in both encoding and decoding. The GPU memory usage of DeepRAHT is only 8 GB, suggesting that it can be much faster if it increases the number of points in the blocks. Notably, our model only needs a single checkpoint with 88 MB, maintains a moderate model size. This efficiency is attributed to our relatively shallow network architecture and the efficient sparse convolution, which provides a significant advantage for practical applications.

Method	3DAC	TSC-PCAC	Unicorn	DeepRAHT
Enc. Time	38.45 s	7.86 s	20.86 s	**6.03 s**
Dec. Time	51.71 s	26.87 s	14.99 s	**5.74 s**
Model Size	**1 MB×5**	148 MB×5	65 MB×3	88 MB×1
GPU Mem.	10 GB	22 GB	16 GB	**8 GB**

Table 3: Complexity comparison tested on 8iVSLF.

Ablation Study The results of the ablation experiments on loot_viewdep are presented in Fig. 8. The `RAHT` represents the vanilla RAHT proposed in (De Queiroz and Chou 2016) without any prediction, which is also the version used by 3DAC. `RAHT+Pred` refers to DeepRAHT only utilizing the IDW prediction and achieves approximately 48.2% BD rate savings compared to the vanilla `RAHT`. `RAHT+Pred` has the same structure and algorithm as `G-PCCv14`; therefore, its performance is very close to that of `G-PCCv14`, which serves as the lower bound of DeepRAHT, ensuring its robustness. DeepRAHT with the prediction compensation module, denoted as `RAHT+Pred+Comp`, further obtained a 24.6% BD rate gain compared to `G-PCCv14`. Additionally, DeepRAHT outperforms `G-PCCv23` by 16.6% without requiring sibling context in prediction, as discussed in the **Prediction Model** section. These results demonstrate the effectiveness of the prediction model and the prediction compensation module of DeepRAHT.

Conclusion

In this paper, we study the learning of prediction in RAHT. We implemented an end-to-end differentiable predictive RAHT called DeepRAHT, enabling the entire framework to be trained jointly. We propose a learning-based in-loop prediction model that leverages context from the father and grandfather scales, which obtained significant performance gains. We also design a rate proxy based on the Laplace distribution, which has been proven useful for estimating the bitrate of run-length coding and is more efficient and robust for coding RAHT coefficients. DeepRAHT achieves a 16% bitrate saving compared to G-PCCv23 and outperforms existing learning-based compression methods across multiple datasets, showcasing significant advantages in high performance, low complexity, and exceptional robustness. Future work will focus on extending this approach to the compression of LiDAR and dynamic point clouds.

Acknowledgements

This work is partially supported by the Research Grant Council (RGC) of Hong Kong General Research Fund (GRF) under Grant 11200323, the NSFC/RGC JRS Project N_CityU198/24, and ITC grant GHP/044/21SZ.

References

Alexiou, E.; Tung, K.; and Ebrahimi, T. 2020. Towards neural network approaches for point cloud compression. In *Applications of digital image processing XLIII*, volume 11510, 18–37. SPIE.

Ballé, J.; Minnen, D.; Singh, S.; Hwang, S. J.; and Johnston, N. 2018. Variational image compression with a scale hyperprior. In *International Conference on Learning Representations*.

Bengio, Y.; Léonard, N.; and Courville, A. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

Choy, C.; Gwak, J.; and Savarese, S. 2019. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 3075–3084.

Cohen, R. A.; Tian, D.; and Vetro, A. 2016. Attribute compression for sparse point clouds using graph transforms. In *2016 IEEE International Conference on Image Processing (ICIP)*, 1374–1378.

De Queiroz, R. L.; and Chou, P. A. 2016. Compression of 3D point clouds using a region-adaptive hierarchical transform. *IEEE Transactions on Image Processing*, 25(8): 3947–3956.

Fang, G.; Hu, Q.; Wang, H.; Xu, Y.; and Guo, Y. 2022. 3dac: Learning attribute compression for point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14819–14828.

Fu, C.; Li, G.; Song, R.; Gao, W.; and Liu, S. 2022. Octattention: Octree-based large-scale contexts model for point cloud compression. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, 625–633.

Graziosi, D.; Nakagami, O.; Kuma, S.; Zaghetto, A.; Suzuki, T.; and Tabatabai, A. 2020. An overview of ongoing point cloud compression standardization activities: Video-based (V-PCC) and geometry-based (G-PCC). *APSIPA Transactions on Signal and Information Processing*, 9: e13.

Guo, Z.; Zhang, Y.; Zhu, L.; Wang, H.; and Jiang, G. 2024. TSC-PCAC: Voxel Transformer and Sparse Convolution-Based Point Cloud Attribute Compression for 3D Broadcasting. *IEEE Transactions on Broadcasting*.

Isik, B.; Chou, P. A.; Hwang, S. J.; Johnston, N.; and Toderici, G. 2022. Lvac: Learned volumetric attribute compression for point clouds using coordinate based networks. *Frontiers in Signal Processing*, 2: 1008812.

Jonathan Taquet, S. L. 2020. Report on dyadic RAHT. Technical Report m54266, ISO/IEC JTC1/SC29/WG11 MPEG, BlackBerry.

Kerbl, B.; Kopanas, G.; Leimkühler, T.; and Drettakis, G. 2023. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4): 139–1.

Krivokuća, M.; Chou, P. A.; and Savill, P. 2018. 8i Voxelized surface light field (8iVSLF) dataset. Input document m42914, ISO/IEC JTC1/SC29/WG11 MPEG, Ljubljana.

Li, L.; Li, Z.; Liu, S.; and Li, H. 2020. Efficient projected frame padding for video-based point cloud compression. *IEEE Transactions on Multimedia*, 23: 2806–2819.

Li, X.; Dai, W.; Li, S.; Li, C.; Zou, J.; and Xiong, H. 2024. 3-D Point Cloud Attribute Compression With p-Laplacian Embedding Graph Dictionary Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(2): 975–993.

Mao, X.; Yuan, H.; Guo, T.; Jiang, S.; Hamzaoui, R.; and Kwong, S. 2025. SPAC: Sampling-Based Progressive Attribute Compression for Dense Point Clouds. *IEEE Transactions on Image Processing*, 34: 2939–2953.

Mekuria, R.; Blom, K.; and Cesar, P. 2016. Design, implementation, and evaluation of a point cloud codec for tele-immersive video. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(4): 828–842.

MPEG 3DG. 2016. Use Cases for Point Cloud Compression (PCC). Technical Report N16331, ISO/IEC JTC1/SC29/WG11 MPEG.

MPEG 3DG. 2020. Report on modifying entropy coding of attributes coefficients with dictionary removal and run-length coding. Technical Report m54265, ISO/IEC JTC1/SC29/WG11 MPEG.

MPEG 3DG. 2022. Common test conditions for G-PCC. Technical Report N0427, ISO/IEC JTC 1/SC 29/WG 7 MPEG.

MPEG 3DG. 2023. G-PCC 2nd Edition codec description. Technical Report N00506, ISO/IEC JTC1/SC29/WG7 MPEG.

MPEG 3DG. 2024. CTC on AI-based point cloud compression. Technical Report N01058, ISO/IEC JTC1/SC29/WG7 MPEG.

Nguyen, D. T.; and Kaup, A. 2023. Lossless point cloud geometry and attribute compression using a learned conditional probability model. *IEEE Transactions on Circuits and Systems for Video Technology*, 33(8): 4337–4348.

Quach, M.; Valenzise, G.; and Dufaux, F. 2020. Folding-based compression of point cloud attributes. In *2020 IEEE International Conference on Image Processing (ICIP)*, 3309–3313. IEEE.

Schwarz, S.; Preda, M.; Baroncini, V.; Budagavi, M.; Cesar, P.; Chou, P. A.; Cohen, R. A.; Krivokuća, M.; Lasserre, S.; Li, Z.; et al. 2018. Emerging mpeg standards for point cloud compression. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(1): 133–148.

Shao, Y.; Zhang, Z.; Li, Z.; Fan, K.; and Li, G. 2017. Attribute compression of 3D point clouds using laplacian sparsity optimized graph transform. In *2017 IEEE Visual Communications and Image Processing (VCIP)*, 1–4.

Sheng, X.; Li, L.; Liu, D.; Xiong, Z.; Li, Z.; and Wu, F. 2022. Deep-PCAC: an end-to-end deep lossy compression framework for point cloud attributes. *IEEE Transactions on Multimedia*, 24: 2617–2632.

Song, F.; Li, G.; Yang, X.; Gao, W.; and Liu, S. 2023a. Block-Adaptive Point Cloud Attribute Coding With Region-Aware Optimized Transform. *IEEE Transactions on Circuits and Systems for Video Technology*, 33(8): 4294–4308.

Song, R.; Fu, C.; Liu, S.; and Li, G. 2023b. Efficient hierarchical entropy model for learned point cloud compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14368–14377.

Sébastien Lasserre, D. F. 2019. On an improvement of RAHT to exploit attribute correlation. Technical Report m47378, ISO/IEC JTC1/SC29/WG11 MPEG, BlackBerry.

Wang, J.; Ding, D.; and Ma, Z. 2023. Lossless point cloud attribute compression using cross-scale, cross-group, and cross-color prediction. In *2023 Data Compression Conference (DCC)*, 228–237.

Wang, J.; and Ma, Z. 2022. Sparse tensor-based point cloud attribute compression. In *2022 IEEE 5th International Conference on Multimedia Information Processing and Retrieval (MIPR)*, 59–64. IEEE.

Wang, J.; Xue, R.; Li, J.; Ding, D.; Lin, Y.; and Ma, Z. 2025a. A Versatile Point Cloud Compressor Using Universal Multiscale Conditional Coding – Part I: Geometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(1): 269–287.

Wang, J.; Xue, R.; Li, J.; Ding, D.; Lin, Y.; and Ma, Z. 2025b. A Versatile Point Cloud Compressor Using Universal Multiscale Conditional Coding – Part II: Attribute. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(1): 252–268.

Wang, K.; and Gao, W. 2025. Unipcgc: Towards practical point cloud geometry compression via an efficient unified approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 12721–12729.

Wang, W.; Xu, Y.; Zhang, K.; and Zhang, L. 2023. Peer Upsampled Transform Domain Prediction for G-PCC. In *2023 IEEE International Conference on Multimedia and Expo (ICME)*, 708–713.

Waschbüsch, M.; Gross, M. H.; Eberhard, F.; Lamboray, E.; and Würmlin, S. 2004. Progressive Compression of Point-Sampled Models. In *PBG*, 95–102.

Xu, Y.; Hu, W.; Wang, S.; Zhang, X.; Wang, S.; Ma, S.; Guo, Z.; and Gao, W. 2020. Predictive generalized graph fourier transform for attribute compression of dynamic point clouds. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(5): 1968–1982.

Xu, Y.; Lu, Y.; and Wen, Z. 2017. Owlii Dynamic human mesh sequence dataset. Input document m41658, ISO/IEC JTC1/SC29/WG11 MPEG, Macau.

You, K.; Chen, T.; Ding, D.; Asif, M. S.; and Ma, Z. 2025. Reno: Real-time neural compression for 3d lidar point clouds. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 22172–22181.

Yu, P.; Zhang, Y.; Liang, F.; Li, H.; and Guo, Y. 2025. Hierarchical Distortion Learning for Fast Lossy Compression of Point Clouds. *IEEE Transactions on Multimedia*, 1–16.

Zhang, J.; Chen, Y.; Liu, G.; Gao, W.; and Li, G. 2024. Efficient Point Cloud Attribute Compression Framework using Attribute-Guided Graph Fourier Transform. In *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 8426–8430.