Inter-Frame Compression for Dynamic Point Cloud Geometry Coding

Anique Akhtar, *Member, IEEE*, Zhu Li, *Senior Member, IEEE*, and Geert Van der Auwera, *Senior Member, IEEE*

***Abstract*— Efficient point cloud compression is essential for applications like virtual and mixed reality, autonomous driving, and cultural heritage. This paper proposes a deep learning-based inter-frame encoding scheme for dynamic point cloud geometry compression. We propose a lossy geometry compression scheme that predicts the latent representation of the current frame using the previous frame by employing a novel feature space inter-prediction network. The proposed network utilizes sparse convolutions with hierarchical multiscale 3D feature learning to encode the current frame using the previous frame. The proposed method introduces a novel predictor network for motion compensation in the feature domain to map the latent representation of the previous frame to the coordinates of the current frame to predict the current frame's feature embedding. The framework transmits the residual of the predicted features and the actual features by compressing them using a learned probabilistic factorized entropy model. At the receiver, the decoder hierarchically reconstructs the current frame by progressively rescaling the feature embedding. The proposed framework is compared to the state-of-the-art Video-based Point Cloud Compression (V-PCC) and Geometry-based Point Cloud Compression (G-PCC) schemes standardized by the Moving Picture Experts Group (MPEG). The proposed method achieves more than 88% BD-Rate (Bjøntegaard Delta Rate) reduction against G-PCCv20 Octree, more than 56% BD-Rate savings against G-PCCv20 Trisoup, more than 62% BD-Rate reduction against V-PCC intra-frame encoding mode, and more than 52% BD-Rate savings against V-PCC P-frame-based inter-frame encoding mode using HEVC. These significant performance gains are cross-checked and verified in the MPEG working group.**

***Index Terms*— Point cloud, compression, PCC, deep learning, neural network.**

I. Introduction

A POINT cloud (PC) is a 3D data representation that is essential for tasks like virtual reality (VR) and mixed reality (MR), autonomous driving, cultural heritage, etc. PCs are a set of points in 3D space, represented by their 3D coordinates (*x, y, z*) referred to as the *geometry*. Each point may also be associated with multiple *attributes* such as color, normal vectors, and reflectance. Depending on the target application and the PC acquisition methods, the PC can be categorized into point cloud scenes and point cloud objects. Point cloud scenes are typically captured using LiDAR sensors and are often dynamically acquired. Point cloud objects can be further subdivided into static point clouds and dynamic point clouds. A static PC is a single object, whereas a dynamic PC is a time-varying PC where each instance of a dynamic PC is a static PC. Dynamic time-varying PCs are used in AR/VR, volumetric video streaming, and telepresence and can be generated using 3D models, i.e., CGI, or captured from real-world scenarios using various methods such as multiple cameras with depth sensors surrounding the object. These PCs are dense photo-realistic point clouds that can have a massive amount of points, especially in high precision or large-scale captures (millions of points per frame with up to 60 frames per second (FPS)). Therefore, efficient point cloud compression (PCC) is particularly important to enable practical usage in VR and MR applications. This paper focuses on geometry compression for the dense dynamic point clouds. Temporally successive point cloud frames share some similarities, motion estimation is key to effective compression of these sequences. However, these frames may have different numbers of points, and exhibit no explicit association between points over time. Performing motion estimation, motion compensation, and effective compression of such data is, therefore, a challenging task.

The Moving Picture Experts Group (MPEG) has approved two PCC standards [1], [2]: Geometry-based Point Cloud Compression (G-PCC) [3] and Video-based Point Cloud Compression (V-PCC) [4]. G-PCC includes octree-geometry coding as a generic geometry coding tool and a predictive geometry coding (tree-based) tool which is more targeted toward LiDAR-based point clouds. G-PCC is still developing a triangle meshes or triangle soup (trisoup) based method to approximate the surface of the 3D model. V-PCC on the other hand encodes dynamic point clouds by projecting 3D points onto a 2D plane and then uses video codecs, e.g., High-Efficiency Video Coding (HEVC), to encode each frame over time. MPEG has also proposed common test conditions (CTC) to evaluate test models [5].

Deep learning solutions for image and video encoding have been widely successful [6]. Recently, similar deep learning-based PC geometry compression methods [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18] have been shown

Manuscript received 22 August 2022; revised 17 July 2023, 2 October 2023, and 15 November 2023; accepted 4 December 2023. Date of publication 3 January 2024; date of current version 8 January 2024. This work was supported in part by NSF under Award 1747751 and Award 2148382. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Ioan Tabus. *(Corresponding author: Anique Akhtar.)*

Anique Akhtar and Geert Van der Auwera are with Qualcomm Technologies Inc., San Diego, CA 92121 USA (e-mail: aniquea@qti.qualcomm.com; geertv@qti.qualcomm.com).

Zhu Li is with the Department of Computer Science and Electrical Engineering, University of Missouri-Kansas City, Kansas City, MO 64110 USA (e-mail: zhu.li@ieee.org).

Digital Object Identifier 10.1109/TIP.2023.3343096

to provide significant coding gains over traditional methodologies. Point cloud compression represents new challenges due to the unique characteristics of PC. For instance, the unstructured representation of PC data, the sparse nature of the data, as well as the massive number of points per PC, specifically for dense photo-realistic PC, makes it difficult to exploit spatial and temporal correlation. The current deep learning-based PC geometry compression solutions are all intra-prediction methods for static point clouds and fail to utilize inter-prediction coding gains by predicting the current frame using previously decoded frames.

Inter-prediction schemes in video compression are very successful in performing motion compensation to achieve impressive results. However, similar motion compensation for dynamic point clouds is not possible because the coordinates between different frames of a point cloud sequence are different due to non-uniform sampling in the spatial-temporal space of the point cloud geometry. Performing motion estimation across frames with changing voxels occupancy is challenging and hence the deep-learning solutions struggle to perform motion compensation on dynamic point cloud frames. To this end, we propose a novel inter-frame point cloud compression scheme that successfully performs motion compensation. Following MPEG's PCC category guidelines, our work seeks to target dense dynamic point clouds used for VR/MR and immersive telecommunications. Sparse dynamically acquired LiDAR-based point clouds are a very different point cloud category that is out of the scope of this work. Our contributions are summarized as follows:

- A novel deep learning-based framework is proposed for point cloud geometry inter-frame encoding similar to P-frame encoding in video compression.
- We propose a novel inter-prediction module (predictor network) that learns a feature embedding of the current PC frame from the previous PC frame. The network utilizes hierarchical multiscale feature extractions and employs a generalized sparse convolution (*GSConv*) with arbitrary input and output coordinates to perform motion compensation in the feature domain by mapping the latent features from the coordinates of the first frame to the coordinates of the second frame. The inter-prediction module is the first deep learning module that successfully enables the effective transferring of features between point cloud frames with different coordinates.

Experimental results show the proposed method achieving more than 88% BD-Rate gains against G-PCCv20 (octree), more than 56% BD-Rate gains against G-PCC (trisoup), more than 34% BD-Rate gains against state-of-the-art deep learning-based point cloud geometry compression method, more than 62% BD-Rate gains against V-PCCv18 intra-frame mode, and more than 52% BD-Rate gains against V-PCCv18 P-frame-based (low-delay) inter-frame mode which uses HEVC.

II. Background

Our research is most closely related to three research topics: point cloud geometry compression, deep learning-based video inter-frame coding, and deep learning-based point cloud compression.

Prior non-deep learning-based point cloud geometry compression mostly includes *octree-based, triangle mesh-based*, and *3D-to-2D projection-based* methodologies. **Octree-based methods** are the most widely used point cloud encoding methods [19], [20], [21]. Octree provides an efficient way to partition the 3D space to represent point clouds and is especially suitable for lossless coding. In these methods, the volumetric point cloud is recursively divided into octree decomposition until it reaches the leaf nodes. Then the occupancy of these nodes can be compressed through an entropy context modeling conditioned on neighboring and parent nodes. Thanou et al. [22] implemented octree-based encoding for time-varying point clouds that can predict graph-encoded octree structures between adjacent frames. MPEG's G-PCC standard [1] also employs an octree-based compression method known as *octree geometry codec* and is specifically devoted to sparse point clouds. G-PCC encoding can further be complemented by triangle meshes (a.k.a., triangle soups) which are locally generated together with the octree to terminate the octree decomposition prematurely. This helps reconstruct object surfaces with finer spatial details and is known as the *trisoup geometry codec* [23].

3D-to-2D projection-based methods. Traditional 2D image and video coding have demonstrated outstanding efficiency and have been widely used in standards which have motivated works to project 3D objects to multiple 2D planes and leverage popular image and video codecs for compact representation. MPEG's V-PCC [2] standard is one such 3D-to-2D projection-based solution that is specifically designed for dense, as well as, dynamic PCs. The V-PCC standard projects the points and the corresponding attributes onto planes and then uses a state-of-the-art video codec, such as HEVC, to encode point clouds. V-PCC has both intra-frame coding as well as inter-frame coding [24] where the previously decoded frames are employed to encode the next frames. We have recently also had some works for **dynamic point cloud compression** [25], [26], [27]. However, their results are still lacking and the performance is not comparable to V-PCC.

Deep learning-based models for image and video encoding can learn an optimal non-linear transform from data along with the probabilities required for entropy coding the latent representation into a bitstream in an end-to-end fashion. For image compression, autoencoders [28] were initially adopted and the best results were achieved by employing variational autoencoders with side information transmission and applying an autoregressive model [29]. Deep learning solutions for video compression methods usually employ 3D autoencoders, frame interpolation, and/or motion compensation via optical flow. 3D autoencoders are an extension of deep learned image compression. Frame interpolation methods use neural networks to temporally interpolate between frames in a video and then encode the residuals [30]. Motion compensation via optical flow is based on estimating and compressing optical flow which is applied with bilinear warping to a previously decoded frame to obtain a prediction of the frame currently being encoded [31]. Current deep learning-based PCC takes inspiration from the deep learning-based image compression methods but so far has not been able to implement inter-frame

prediction models commonly used in video encoding. Our work is the first method that takes inspiration from the frame interpolation-based methods in video encoding to perform inter-frame encoding for dynamic point clouds.

Deep learning-based Geometry PCC can be broadly categorized into: *voxelization-based methods, octree-based methods, point-based methods*, and *sparse tensors-based methods*. **Voxelization-based methods** were employed in the earlier approaches, including Quach et al. [7], Wang et al. [8], Guarda et al. [9] and Quach et al. [10]. These methods voxelize the PC and then divide it into smaller blocks typically of $64 \times 64 \times 64$ voxels. Then 3D convolutions are applied using autoencoder architectures to compress these blocks into latent representations. These methods usually employ a focal loss or a weighted binary cross-entropy loss to train their model. However, these methods also have to process empty voxels which are usually the majority of the voxels and are, therefore, computational and memory inefficient.

Octree-based deep learning methods employ octree representation to encode the PCs leading to better consumption of storage and computation. These methods employ entropy context modeling to predict each node's occupancy probability conditioned on its neighboring and parent nodes. MuSCLE [11] and OctSqueeze [12] employ Multi-Layer Perceptrons (MLPs) to exploit the dependency between parent and child nodes. VoxelContext-Net [13] employs both neighbors and parents as well as voxelized neighborhood points as context for probability approximation. Recently, OctAttention [14] has been introduced that increases the receptive field of the context model by employing a large-scale transformer-based context attention module to estimate the probability of occupancy code. All of these methods encode the point cloud in a lossless manner and show promising results, particularly on sparse LiDAR-based point clouds.

Point-based methods directly process raw point cloud data without changing their representation or voxelizing them. They typically employ PointNet [32] or PointNet++ [33] type architectures that process raw point clouds using point-wise fully connected layers. These methods are typically patch-based methods that employ farthest point sampling to subsample and a knn search to find per point feature embedding to build an MLP-based autoencoder. However as seen in some of these works [15], [16], [34], the coding efficiency of such point-wise models is still relatively low and fails to generalize to large-scale dense point clouds. Furthermore, these methods require a lot of pre and post-processing making the encoding process computationally inefficient.

Recent **sparse convolution-based methods** [17], [18], [35] have shown really good results especially for denser photo-realistic point clouds. Sparse convolutions exploit the inherent sparsity of point cloud data for complexity reduction allowing for very large point clouds to be processed by a deeper sparse convolutional network. This allows the network to better capture the characteristics of sparse and unstructured points and better extraction of local and global 3D geometric features. However, all of these works employ intra-frame encoding for static point clouds. We employ sparse convolution-based autoencoder architecture similar to [17] and design a sparse convolutional inter-frame prediction module that encodes the next PC frame using the previously decoded PC frame similar to P-frame prediction in video encoding.

III. Proposed Method

The proposed lossy inter-frame point cloud geometry compression framework is illustrated in Fig. 1. We employ sparse tensors and sparse convolutions to decrease the computational complexity of the network so it can process two PC frames. The solution takes inspiration from the PCGCv2 [17] multiscale point cloud geometry compression (PCGC) work. PCGCv2 is an intra-frame point cloud compression scheme suitable for static point clouds. The proposed inter-frame compression framework employs an encoder and decoder network similar to PCGCv2 along with a novel inter-prediction module to predict the feature embedding of the current PC frame from the previous PC frame. The proposed inter-prediction module employs a specific version of generalized sparse convolution [36] with different input and output coordinates denoted as *GSConv* to perform motion estimation in the feature domain. The inter-prediction module is a standalone module that can be employed with different network architectures. The residual between the predicted and ground truth features are calculated and then these residuals along with the three-times downsampled coordinates are transmitted to the receiver. The three-times downsampled coordinates are losslessly encoded by an octree encoder using G-PCC [3], whereas the residual features are encoded in a lossy manner using factorized entropy model to predict the probability distribution for arithmetic coding. It should be noted that in our system, the encoder and prediction network is present both at the transmitter as well as the receiver. We train the networks with joint reconstruction and bit-rate loss to optimize rate distortion. We provide a detailed description of all our modules in subsequent discussions.

A. Problem Formulation and Preprocessing

We adopt sparse convolutions for low-complexity tensor processing and build our system using Minkowski Engine [36]. Each point cloud frame is converted into a sparse tensor P. Each point cloud tensor $P = \{C_n, F_n\}_n$ is represented by a set of coordinates $C = \{(x_n, y_n, z_n)\}_n$ and their associated features $F = \{f(x_n, y_n, z_n)\}_n$. Only the occupied coordinates are kept in a sparse tensor. To initialize the input point cloud as geometry only, we assign feature $f(x, y, z) = 1$ to each occupied coordinate. Given a dynamic point cloud with multiple frames, P^i, our goal is to convert them into a latent representation with the smallest possible bitrate. We use P-frame encoding where the current frame is encoded using the prediction from the previous frame. We denote the Encoder network as E, and the Decoder network as D.

B. Feature Extraction

Our encoder and decoder network is shown in Fig. 2. We utilize the Inception-Residual Block (IRB) [37] for feature extraction in all our networks. Each IRB contains three Inception-Residual Network (IRN) similar to PCGCv2 [17].

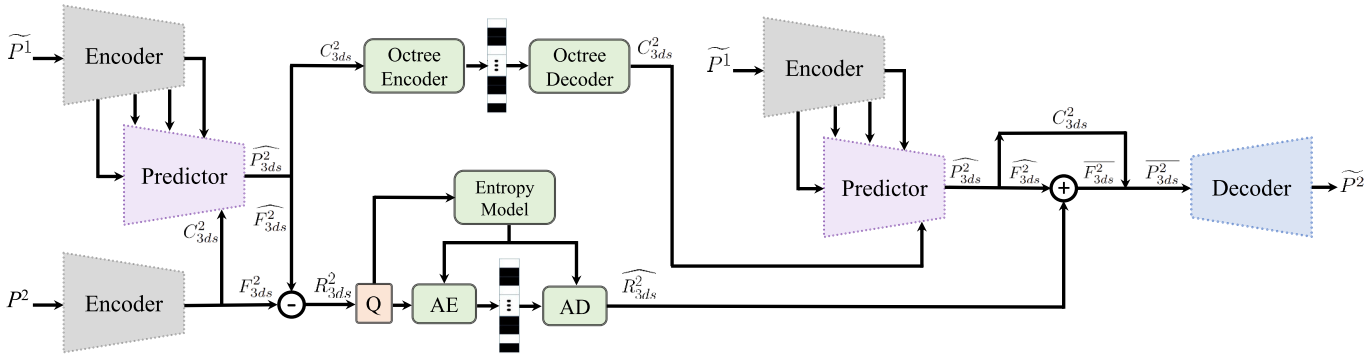


Fig. 1. System model. The previously decoded frame $\widetilde{P^1}$ is employed to encode a feature embedding of the current frame P^2. Multiscale features from $\widetilde{P^1}$ and three-times downsampled coordinates C^2_{3ds} from P^2 are passed to the Predictor network to learn a feature embedding $\widehat{P^2_{3ds}} = \{C^2_{3ds}, \widehat{F^2_{3ds}}\}$. The current frame's three-times downsampled coordinates C^2_{3ds} are transmitted in a lossless manner using an octree encoder. The predicted downsampled features $\widehat{F^2_{3ds}}$ and the original downsampled features F^2_{3ds} are subtracted to obtain the residual features R^2_{3ds}. The residual is transmitted in a lossy manner using a learned entropy model. The same Encoder and Predictor module are used throughout the system. Q, AE, and AD stand for quantization, arithmetic encoder, and arithmetic decoder respectively.

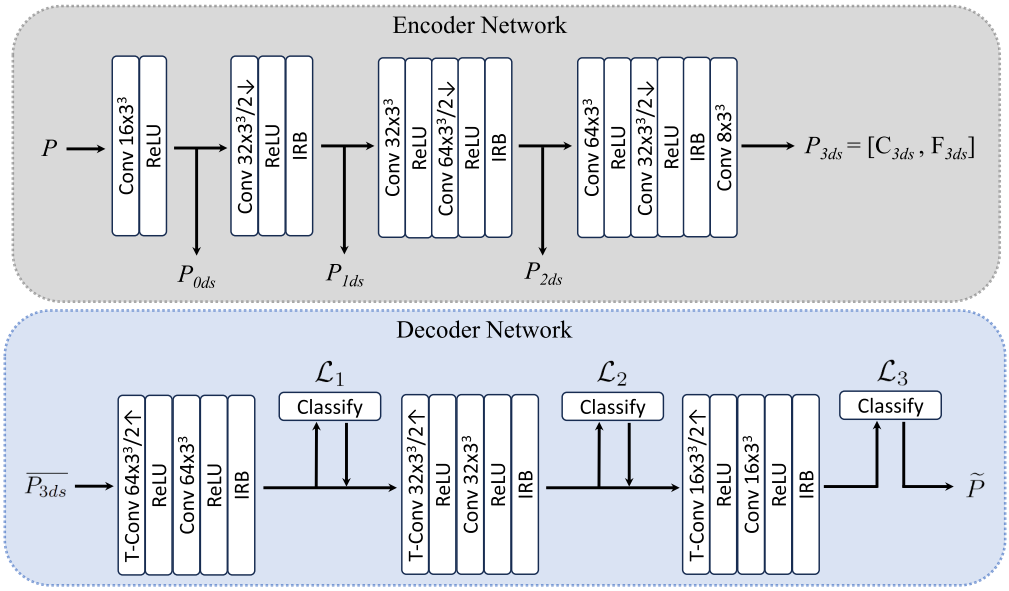


Fig. 2. Encoder and decoder network. The encoder network takes the original point cloud sparse tensor P, and creates sparse features at four different scales: P_{0ds}, P_{1ds}, P_{2ds}, and P_{3ds}. Where P_{3ds} denotes three-times downsampled sparse tensor containing both the coordinates C_{3ds} and their respective features F_{3ds}. The decoder network takes the three-times downsampled sparse tensor and hierarchically reconstructs the original point cloud by progressively rescaling. The decoder upsamples the sparse tensor one scale at a time using transpose convolution followed by classification and pruning to prune out the false voxels.

We employ a multiscale re-sampling with downscaling at the encoder and upscaling at the decoder. This helps exploit the sparsity of the PC while encoding 3D geometric structural variations into feature attributes of the latent representation. The encoder is used as a feature extraction module to obtain PC tensors at four different scales capturing multiscale features at different level of details: $P_{0ds}, P_{1ds}, P_{2ds}, P_{3ds} = E(P)$. Where P_{jds} represents a sparse tensor P that has been down-sampled j times.

C. Point Cloud Reconstruction

The decoder receives a three-times downsampled PC tensor and upsamples it hierarchically to reconstruct the original PC tensor by employing a different reconstruction loss at each scale: $\widetilde{P} = D(\widehat{P_{3ds}})$. Decoder employs transpose convolution to upsample the PC tensor and generate newer voxels. After each upsampling, the probability of voxel occupancy p_v is predicted and a binary classification loss is employed. The geometry at the decoder is reconstructed by employing a classification and pruning layer to prune false voxels and extract true occupied voxels using binary classification after each upscaling. We employ binary cross-entropy loss for voxel occupancy classification as the distortion loss in each pruning layer at the decoder:

$$\mathcal{L}_{BCE} = \frac{1}{N}\sum_{v} -(\mathcal{O}_v \log p_v + (1-\mathcal{O}_v)\log(1-p_v)) \quad (1)$$

where O_v is the ground truth of whether the voxel v is occupied (1) or unoccupied (0).

One example of a classification and pruning layer is shown in Fig. 3. In this example, the input sparse tensor P_a has

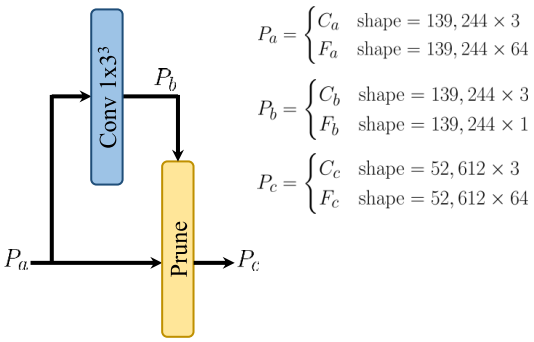


Fig. 3. Example of classification and pruning layer with input sparse tensor P_a and output sparse tensor P_c. Binary classification is applied to P_b to chose the top voxels and prune false voxels from P_a to obtain P_c.

coordinates C_a of shape $139{,}244 \times 3$ and their corresponding features of shape $139{,}244 \times 64$. We pass P_a through a convolution of channel size 1 to obtain sparse tensor P_b with features F_b of shape $139{,}244 \times 1$. From F_b we select the top k features (in this example $k = 52{,}612$) with the highest probability of occupancy (p_v) and their corresponding coordinates using binary voxel classification. The false coordinates and their corresponding features are then pruned from P_a to obtain P_c. The binary cross entropy loss is employed at each of the pruning layers and is applied to tensor P_b so the true voxels could have a higher p_v and the false voxels have a lower p_v. Then the top k voxels with the highest probability of occupancy (p_v) are chosen and the rest of the false voxels are pruned out. k is a metadata calculated at the encoder and losslessly transmitted to the receiver with a very small overhead along with other overhead bits. k determines the number of occupied voxels at each scale for that particular frame which is employed during point cloud reconstruction at the decoder.

D. Overall System Model

This subsection intends to explain the working of the overall system framework shown in Fig. 1. In our work, we denote the current PC frame as P^2 while the previously decoded PC frame is denoted by $\widetilde{P^1}$. The same encoder and prediction module are used throughout the system to decrease the number of parameters. Previously decoded frame $\widetilde{P^1}$ is passed through the encoder to obtain multiscale tensors P^1_{0ds}, P^1_{1ds}, P^1_{2ds}, P^1_{3ds}. The current frame (P^2) is also passed through the encoder to obtain three-times downsampled tensor containing coordinates and features: $P^2_{3ds} = \{C^2_{3ds}, F^2_{3ds}\}$. Current frame's three-times downsampled coordinates (C^2_{3ds}) and the multiscale features from the previous frame are passed to the prediction network to obtain current frame's predicted three-times downsampled tensor $\widehat{P^2_{3ds}} = \{C^2_{3ds}, \widehat{F^2_{3ds}}\}$. The predicted downsampled features $\widehat{F^2_{3ds}}$ and the original downsampled features F^2_{3ds} are subtracted to obtain the residual features R^2_{3ds}. The residual is transmitted in a lossy manner using a factorized entropy model [28]. The current frame's three-times downsampled coordinates C^2_{3ds} are transmitted in a lossless manner using an octree encoder like G-PCC [3]. Three-times downsampled coordinates C^2_{3ds} is much smaller than the original geometry (e.g. for the 8iVFB dataset [38], the C^2_{3ds} is about 16 times smaller than C^2). At the receiver, the previously decoded frame $\widetilde{P^1}$ and the three-times downsampled coordinates C^2_{3ds} are used to predict $\widehat{P^2_{3ds}}$. The residual $\widehat{R^2_{3ds}}$ is added with $\widehat{P^2_{3ds}}$ to obtain the current frame's three-times downsampled tensor representation $\widetilde{P^2_{3ds}}$. The decoder progressively rescales $\widetilde{P^2_{3ds}}$ to obtain the current decoded frame $\widetilde{P^2}$. Encoder and decoder architecture can on their own be used without the prediction module for intra-frame PC compression.

E. Inter-Prediction Module

Until now, efficient motion estimation for dynamic point clouds has not been possible due to the difference in the occupied coordinates between point cloud frames. We propose a novel deep learning-based inter-frame predictor network that can predict the latent representation of the current frame from the previously reconstructed frame as shown in Fig. 4. This is the first inter-prediction module that is capable of performing motion estimation of 3D coordinates across frames such as to learn the current frame's feature embedding. The framework does not explicitly performs motion estimation of distinct 3D points across multiple frames. Instead, it learns the appropriate weights to perform motion compensation for 3D points in the feature domain. We do not employ explicit loss function for motion estimation but perform end-to-end training.

The multiscale features from the previous frame, P^1_{0ds}, P^1_{1ds}, P^1_{2ds}, P^1_{3ds}, and the three downsampled coordinates from the current frame, C^2_{3ds}, are fed to the prediction network to obtain current frame's predicted three-times downsampled tensor $\widehat{P^2_{3ds}} = \{C^2_{3ds}, \widehat{F^2_{3ds}}\}$. The prediction network downscales the input three times while concatenating it with the corresponding scale features. Finally a version of Generalized Sparse Convolution (GSC) is employed to map features from C^1_{3ds} to C^2_{3ds} obtain the tensor $\widehat{P^2_{3ds}}$.

GSC is defined in [36] as a generalized version of sparse convolution that incorporates all discrete convolutions as special cases. Let $x^{\text{in}}_u \in \mathbb{R}^{N^{\text{in}}}$ be an N^{in}-dimensional input feature vector in a D-dimensional space at $u \in \mathbb{R}^D$ (a D-dimensional coordinate), and convolution kernel weights be $W \in \mathbb{R}^{K^D \times N^{\text{out}} \times N^{\text{in}}}$. The conventional dense convolution in D-dimension is defined in [36] as:

$$x^{\text{out}}_u = \sum_{i \in \mathcal{V}^D(K)} W_i x^{\text{in}}_{u+i} \text{ for } u \in \mathbb{Z}^D \tag{2}$$

where $\mathcal{V}^D(K)$ is the list of offsets in D-dimensional hypercube centered at the origin with kernel size K. The generalized sparse convolution is defined in [36] as:

$$x^{\text{out}}_u = \sum_{i \in \mathcal{N}^D(u, C^{\text{in}})} W_i x^{\text{in}}_{u+i} \text{ for } u \in C^{\text{out}} \tag{3}$$

where $\mathcal{N}^D$ is a set of offsets that define the shape of a kernel and $\mathcal{N}^D(u, C^{\text{in}}) = \{i \,|\, u+i \in C^{in}, i \in \mathcal{N}^D\}$ as the set of offsets from the current center, u, that exist in C^{in}. C^{in} and C^{out} are predefined input and output coordinates of sparse tensors. In GSC, the input and output coordinates are not necessarily

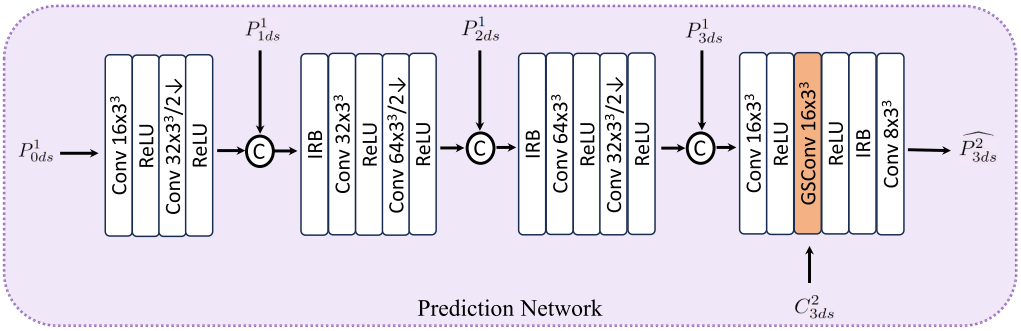
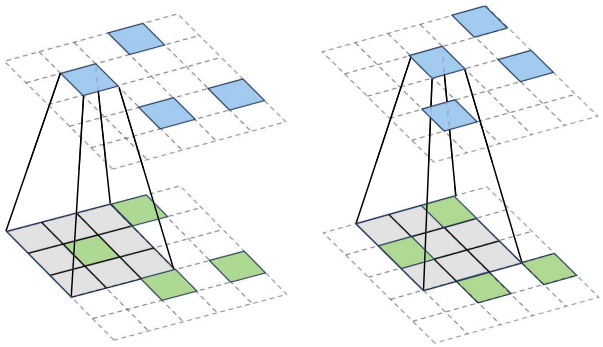


Fig. 4. Prediction network. Takes in four multiscale features from the previous frame and the three-times downsampled coordinates of the current frame (C^2_{3ds}) to learn the current frame's feature embedding $\widehat{P^2_{3ds}}$.



(a) *Conv: Same in/out coords* (b) *GSConv: Arbitrary in/out coords*

Fig. 5. Comparison between the two generalized sparse convolutions employed in the proposed framework. Shown in 2D with blue as the output coordinates (C^{out}) and green as the input coordinates (C^{in}).

the same and the shape of the convolution kernel is arbitrarily defined with $\mathcal{N}^D$.

The proposed framework employs convolution kernel shape ($\mathcal{N}^D$) of a 3×3x3 cuboid. The framework employs two kinds of generalized sparse convolutions which are shown in Fig. 5. Sparse convolutions denoted by *Conv* has the same input (C^{in}) and output (C^{out}) coordinates as shown in Fig. 5a and is employed in encoder, decoder and predictor networks. Sparse convolution denoted by *GSConv* has different input (C^{in}) and output (C^{out}) coordinates as shown in Fig. 5b and is employed only once in the predictor network.

GSConv is employed towards the end of the predictor network to map the features from the input coordinates C^1_{3ds} to the output coordinates C^2_{3ds} while applying a convolution operation. *GSConv* performs motion estimation in the feature domain by translating the latent features from the downsampled coordinates of P^1, i.e., C^1_{3ds} to the downsampled coordinates of P^2, i.e., C^2_{3ds}. This way the *GSConv* enables us to predict learned features for the current frame coordinates using the previous frames multi-scale coordinates.

F. Training

During training, we optimize the Lagrangian loss, i.e.,

$$J_{loss} = R + \lambda D \tag{4}$$

where R is the compressed bit rate and D is the distortion loss. We employ three binary cross-entropy losses at three different scales such that the total distortion loss is:

$$D = \mathcal{L}_1(\widetilde{P}_{2ds}, P_{2ds}) + \mathcal{L}_2(\widetilde{P}_{1ds}, P_{1ds}) + \mathcal{L}_3(\widetilde{P}, P) \tag{5}$$

where the ground-truth P_{2ds} and P_{1ds} are obtained by voxel or quantization-based downsampling of the original point cloud P.

The three downsampled coordinates C^2_{3ds} are transmitted losslessly using Octree encoder in G-PCC [3] and consumes a very small amount of bits (i.e., around 0.024 bpp for 8iVFB dataset [38]). We subtract the three downsampled predicted features $\widehat{F^2_{3ds}}$ from the original three downsampled features F^2_{3ds} to obtain the residual features R^2_{3ds}.

The residual features R^2_{3ds} are quantized before encoding. Note that the quantization operation is non-differentiable, thus during training, we approximate the quantization process by adding a uniform noise $\mu \sim \mathcal{U}(-0.5, 0.5)$. Quantized residual features (lets call them $\bar{f}$) are encoded by an arithmetic encoder using a fully factorized probabilistic entropy model [29] to estimate the probability distribution of $\bar{f}$, i.e., $p_{\bar{f}|\phi}(\bar{f}|\phi)$, where ϕ are the learnable parameters. Then the bpp of encoding $\bar{f}$ is approximated as:

$$\mathcal{R} = \frac{1}{N} \sum_i \log_2 p_{\bar{f}|\phi^{(i)}}(\bar{f}|\phi^{(i)}) \tag{6}$$

where N is the number of points, and i is the index of channels.

IV. Experiments

A. Experimental Setup

For a fair comparison, we closely follow MPEG's common test conditions (CTC) [5] and employ the same diverse datasets recommended by MPEG for deep learning-based dynamic point cloud compression. The performance of our framework has already been cross-checked and verified by the MPEG 3DG EE 5.3 working group experts.

1) Training Dataset: We train the proposed model using three sequences *longdress, loot*, and *queen*. Sequences *longdress* and *loot* are from 8i Voxelized Full Bodies dataset (8iVFB v2) [38], while sequence *queen* is from Technicolor (https://www.technicolor.com/fr). Each sequence has 300 frames with a frame rate of 30 fps. Each sequence has a 10-bit precision with around 800,000 to 1,000,000 points

TABLE I
BD-RATE GAINS AGAINST THE STATE-OF-THE-ART METHODS USING D1 DISTORTION MEASUREMENTS

	G-PCC (octree)	G-PCC (trisoup)	PCGCv2 [17]	V-PCC intra	V-PCC inter
basketball	-89.15	-60.28	-32.45	-60.46	-48.82
exercise	-88.77	-64.91	-35.44	-62.08	-48.30
model	-86.25	-56.75	-33.69	-61.93	-51.80
redandblack	-88.99	-48.11	-28.31	-59.33	-55.58
soldier	-90.21	-52.73	-40.16	-66.51	-43.60
Average	-88.77	-56.69	-34.08	-62.69	-52.44

TABLE II
REPORTED ENVIRONMENT/FRAMEWORK VARIABLES

Parameter	Value
GPU Type	RTX 3090 Ti
CPU Type	11th Gen Intel® Core™ i9-11900F
Framework	Pytorch
Operating system	Ubuntu 20.04 LTS
Batch size	5
Loss functions	BCE loss
Learning rate policy	Adam
λ values	$\frac{1}{10}, \frac{1}{9}, \frac{1}{6}, \frac{1}{4}, \frac{1}{2.5}, \frac{1}{1.7}, \frac{1}{1.1}$
No. of parameters	2,033,000
Peak Memory Usage (GPU)	15 GB

per point cloud frame. To decrease computational complexity during training, we divide the PC frames into smaller chunks by applying the same kd-tree partition on two consecutive frames.

2) Evaluation Dataset: We evaluate the performance of the proposed framework on five sequences: *redandblack, soldier, basketball, exercise*, and *model*. Sequences *redandblack* and *soldier* are from 8i Voxelized Full Bodies dataset (8iVFB v2) [38], while sequences *basketball, exercise*, and *model* are from Owlii Dynamic Human Textured Mesh Sequence Dataset [39]. Each sequence has a frame rate of 30 fps. The 10-bit precision datasets were employed in our experiments. Since deep learning-based inter-frame compression schemes process multiple frames at a time and hence have limited GPU memory, it is advised to use a maximum of 10-bit precision point clouds.

3) Training Strategy: We train our network with $\lambda = \frac{1}{10}, \frac{1}{9}, \frac{1}{6}, \frac{1}{4}, \frac{1}{2.5}, \frac{1}{1.5}, 1$. The Adam optimizer is utilized with a learning rate decayed from 0.0008 to 0.00001. We train the model for around 40,000 batches with a batch size of 5. We conduct all the experiments on a GeForce RTX 3090 GPU with 24GB memory.

4) Evaluation Metric: The bit rate is evaluated using bits per point (bpp), and the distortion is evaluated using point-to-point geometry (D1) Peak Signal-to-Noise Ratio (PSNR), and point-to-plane geometry (D2) PSNR. For some point clouds, the normals are not available which are required for D2-PSNR calculation. We employ Open3D's normal estimation using 20 neighboring points by employing covariance analysis. The geometry PSNRs are obtained using MPEG's *pc_error* tool [40]. The peak value p is set as 1023 for all the datasets. We plot rate-distortion curves and calculate the BD-Rate (Bjøntegaard Delta Rate) [41] gains using D1-PSNR over different methods.

B. Experimental Results

Our framework and environment variables are shown in Table II.

1) GoP Structure: In video coding, a group of pictures, or GOP structure, specifies the order in which intra- and inter-frames are arranged. In the experiments for our framework, the intra-frame (I frame) is encoded using PCGCv2 [17] and the inter-frame (P frame) is encoded using the proposed framework. In the results, the I frame is encoded after 32 frames and the rest of the 31 frames are encoded as inter-frames P frame.

2) Baseline Setup: We compare our method to the state-of-the-art deep learning intra-frame encoding PCGCv2 [17], MPEG's G-PCC (octree as well as trisoup) [3] methods, as well as MPEG's video-based V-PCC method (inter and intra-frame encoding) [4]. We utilize G-PCC's latest reference implementation TMC13-v20, and for V-PCC the latest implementation TMC2-v18 which uses the HEVC video codec. V-PCC inter-frame low-delay setting which involves P-frame encoding is employed for a fair comparison to the proposed P-frame encoding method. Two extra points for higher bpp have been added for V-PCC.

3) Performance Evaluation: Table I shows the BD-Rate gains of the proposed method over the state-of-the-art using D1-PSNR. The lower the BD-Rate value, the more the improvement is. Our method achieves significant gains compared to G-PCC with an average of 88.77% BD-Rate gains against G-PCC (octree), 56.69% BD-Rate improvement over G-PCC (trisoup). Compared to the deep learning-based model PCGCv2, we achieve a 34.08% BD-Rate improvement. Compared to the V-PCC, we achieve a 62.69% BD-Rate improvement over intra-frame encoding mode and 52.44% BD-Rate improvement over inter-frame encoding mode. The proposed method outperforms V-PCC inter-frame mode across all rates for dense photo-realistic point clouds. Please note that in a previous version of this publication, we showed a 91.68% BD-Rate gains against G-PCC (octree) and 84.41% BD-Rate improvement over G-PCC (trisoup) but those gains were against TMC13-v14 but now we have updated G-PCC results to TMC13-v20.

The D1-PSNR and D2-PSNR rate-distortion curves are shown in Fig. 6 and Fig. 7 respectively. As can be seen, the proposed method has significant coding gains compared with the deep learning-based model PCGCv2. It should be noted that compared with PCGCv2, our method performs much better at higher PSNR and still performs better than PCGCv2 at lower PSNRs. This is because both the proposed method and PCGCv2 transmit the three downsampled coordinates in a lossless manner and their corresponding features

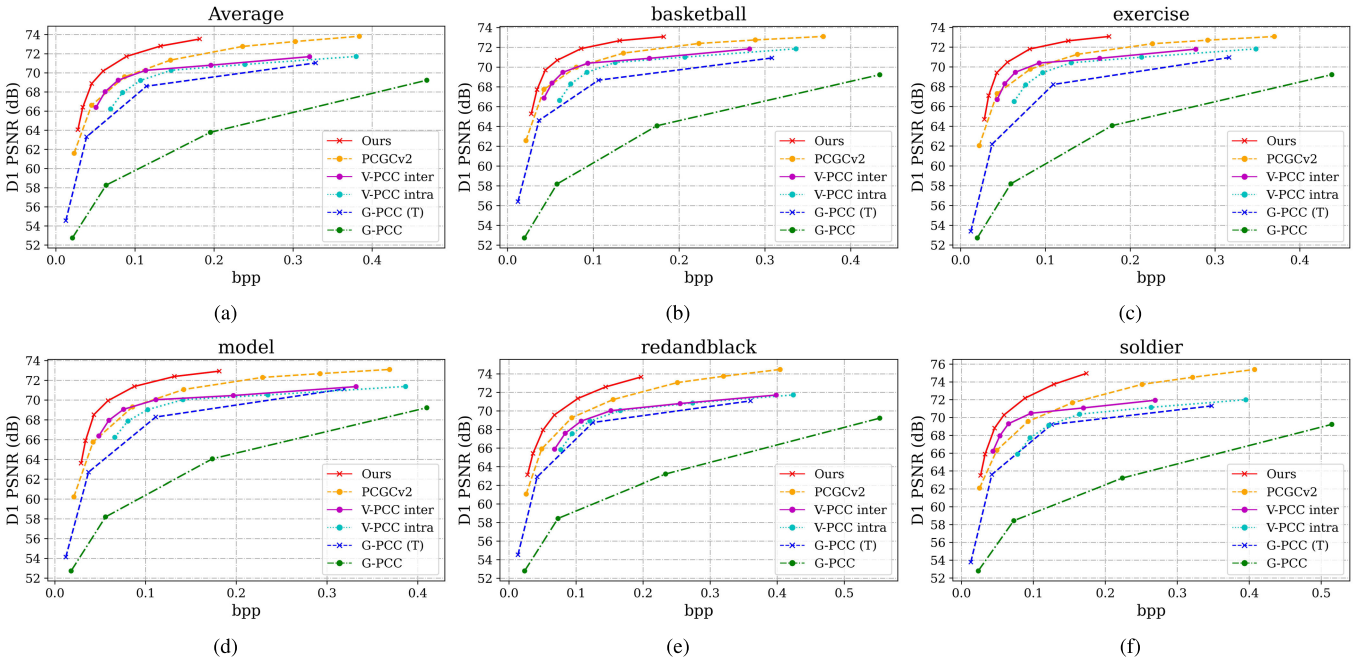


Fig. 6. Rate-distortion curves using D1 PSNR comparison with the state-of-the-arts plotted for five different sequences and their average.

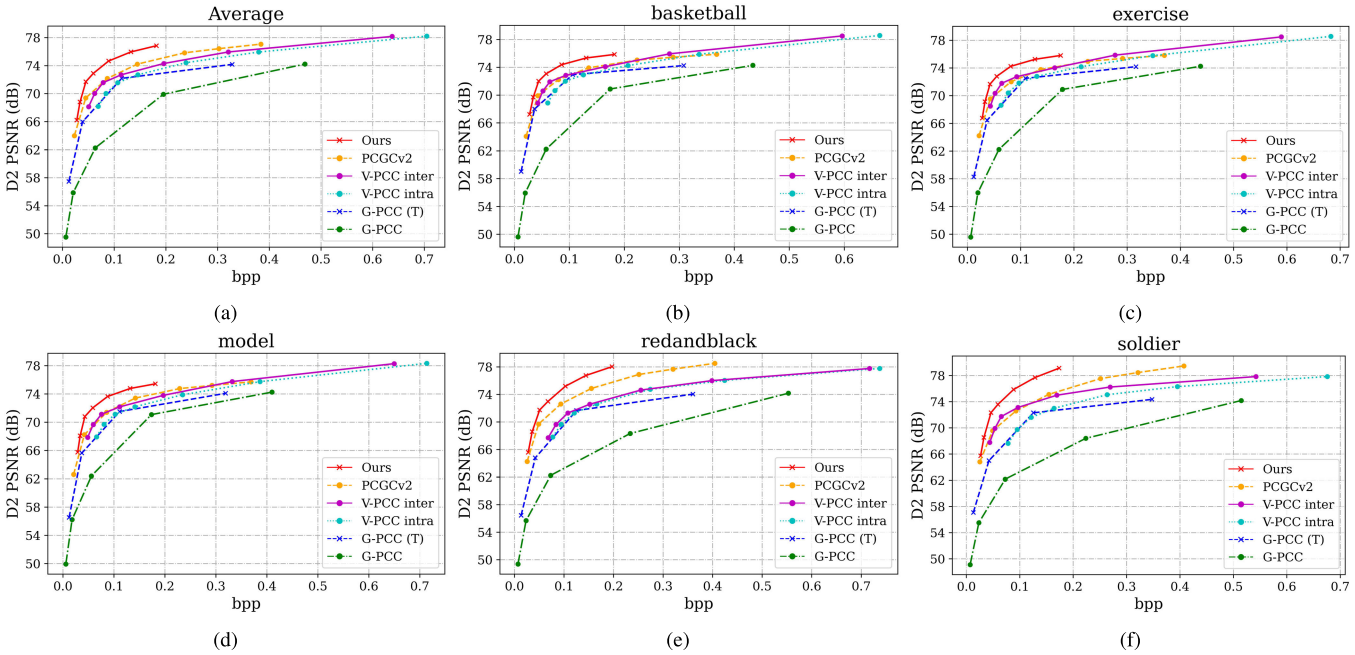


Fig. 7. Rate-distortion curves using D2 PSNR comparison with the state-of-the-arts plotted for five different sequences and their average.

in a lossy manner. However, at lower PSNRs, most of the bits are consumed by coordinates (i.e., around 0.024 bpp) which constitutes the majority of the bitrate. At higher PSNR values most of the bitrates are due to features. Our inter-frame prediction network transmits only the residual of the features and, hence, can significantly decrease the feature bits transmitted leading to much higher gains at higher PSNR and bitrates.

The proposed method also significantly outperforms G-PCC (octree) as well as G-PCC (trisoup). We can notice that G-PCC trisoup performs much better than G-PCC octree which is because trisoup performs better for denser point clouds whereas octree performs better for sparse LiDAR-based point clouds. When compared with V-PCC, we can see that the proposed method achieves a much higher PSNR for the same bitrate for all of the sequences and bitrates. As expected, the V-PCC inter-frame encoding mode performs better than V-PCC intra-frame encoding mode. The sequences that have the most movement (i.e., redandblack) the V-PCC inter and V-PCC intra modes perform pretty similarly whereas the sequence with the least amount of movement (i.e., soldier) the V-PCC inter-frame encoding method performed much

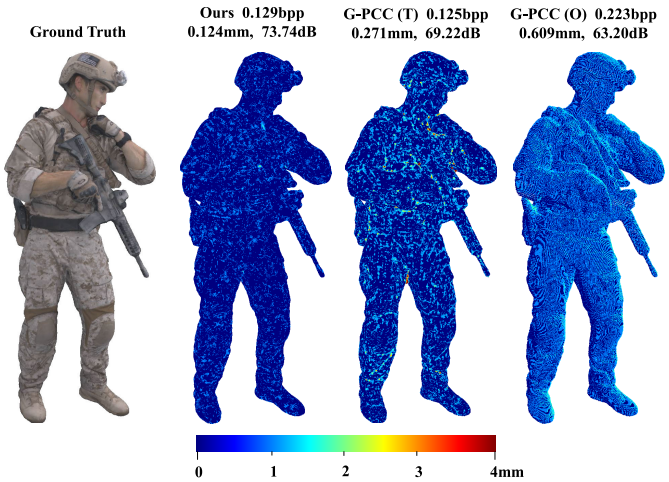


Fig. 8. Qualitative visual comparison of sequence "soldier" for different methods. The color error map describes the point-to-point distortion measured in mm, and the numbers above represent the bitrate, mean error measured in mm, and D1 PSNR.

TABLE III
AVERAGE RUNTIME (PER FRAME) OF DIFFERENT METHODS USING 8iVFBv2 PCs

	G-PCC (O)	G-PCC (T)	PCGCv2 [17]	Ours
Enc (s)	1.50	5.625	0.258	0.364
Dec (s)	0.42	1.61	0.537	0.714

TABLE IV
PARTITIONING THE POINT CLOUD INTO A SMALLER NUMBER OF BLOCKS. TESTED ON SOLDIER SEQUENCE

# of blocks	PSNR	bpp
1	74.56	0.1944
2	74.52	0.1987
4	74.48	0.2055
8	74.35	0.2158

better than V-PCC intra-frame encoding method. We can see a similar pattern between our proposed inter-frame method and PCGCv2 which is an intra-frame method. We see that our proposed inter-frame method has the most improvement over PCGCv2 on the soldier sequence and the least improvement over PCGCv2 on the redandblack sequence. Our Prediction module maps the features extracted from the previous frame to the coordinates extracted from the current frame. In this way, when the motion between adjacent frames is small, the performance is significantly improved.

4) Visual Results: Visual comparison of dense point clouds for geometry only is difficult since it is difficult to differentiate the quality by viewing only the points without color/attribute. The best and most common way to visualize the reconstruction results is to view the per-point distortion error. A qualitative comparison with the proposed method and G-PCC is presented in Fig. 8. Point clouds are colored with the point-to-point reconstruction error for visualization. As can be seen in the visual results too, our method has a much better reconstruction quality compared to G-PCC. Another thing to notice is that the proposed method has very few outliers and generates points very close to the surface of the original point cloud.

C. Runtime Comparison

We compare the runtime of different methods in Table III. We use an Intel Core i9-11900F CPU and an Nvidia GeForce GTX 3090 GPU. G-PCC runtime is computed for the highest bitrate on a CPU. While both PCGCv2 and Our method utilize the GPU. Due to the diversity in platforms, e.g., CPU vs. GPU, Python vs. C/C++, etc, the running time comparison only serves as the intuitive reference to have a general idea about the computational complexity. As can be seen, our method experiences a slight increase in runtime due to processing two PC frames at a time. However, the increased complexity is still minimal given that our network is an inter-frame prediction scheme. PCGCv2 has about 778 thousand parameters, whereas, the proposed method has about 2,033 thousand parameters which is still a relatively small network. The runtime complexity can be optimized by migrating to a C++ implementation and simplifying the framework.

D. Ablation Study: Block Size

Even though in our evaluations, we have used the full point cloud during inference. We wanted to see the effects on PSNR and bitrate of dividing the point cloud into smaller blocks for encoding. The purpose is to demonstrate that if needed, a large point cloud can be partitioned into blocks for processing. During the encoding, we save the *coordinate bitstream, feature bitstream, number of points*, and *the entropy model header information* into four different files. Overall bitrate is decided by the collective size of these files. Once we divide the point cloud into blocks, each block would be encoded separately into four different files so we should expect to see a higher overhead involved. kd-tree partitioning is employed to divide each point cloud into multiple blocks and encoded the blocks independently. The results of this experiment on *soldier* sequence are shown in Table IV. We notice that partitioning the point cloud into smaller blocks decreases the PSNR slightly. However, the difference is minimal. We also notice that the bitrate increases a bit but that could be from the overhead of saving the information in lots of files (e.g. for 8 # of blocks, we have a total of 24 files encoded, whereas, for 1 block, we have a total of 4 files encoded). It is possible to merge these files into a single file to decrease the overhead. However, that is out of the scope of the current work.

V. CONCLUSION

This work proposes a deep learning-based inter-frame compression scheme for dynamic point clouds that encodes the current frame using the decoded previous frame. We employ an encoder to obtain multi-scale features and a decoder to hierarchically reconstruct the point cloud by progressive scaling. The paper introduces a novel inter-prediction module that predicts the latent representation of the current frame by mapping the latent features of the previous frame to the downsampled coordinates of the current frame using a specific version of generalized sparse convolution (*GSConv*) with an arbitrary

input and output coordinates. The proposed method effectively performs motion estimation across frames for dynamic point clouds and encodes and transmits only the residual of the predicted features and the actual features. Sparse convolutions are employed to reduce the space and time complexity which allows the network to process two consecutive point cloud frames per inference. Exhaustive experimental results show more than 88% BD-Rate gains over the state-of-the-art MPEG G-PCC (octree), more than 56% BD-Rate gains over G-PCC (trisoup), more than 34% BD-Rate gains over intra-frame network PCGCv2, more than 62% BD-Rate improvement over MPEG V-PCC intra-frame encoding mode, and more than 52% BD-Rate improvement over MPEG V-PCC inter-frame encoding mode. The proposed method has been verified in MPEG's cross-check.

REFERENCES

[1] S. Schwarz et al., "Emerging MPEG standards for point cloud compression," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 1, pp. 133–148, Mar. 2019.

[2] D. Graziosi, O. Nakagami, S. Kuma, A. Zaghetto, T. Suzuki, and A. Tabatabai, "An overview of ongoing point cloud compression standardization activities: video-based (V-PCC) and geometry-based (G-PCC)," *APSIPA Trans. Signal Inf. Process.*, vol. 9, no. 1, p. e13, 2020.

[3] (2021). *MPEG-PCC-TMC13: Geometry Based Point Cloud Compression G-PCC*. [Online]. Available: https://github.com/MPEGGroup/mpeg-pcc-tmc13

[4] (2022). *MPEG-PCC-TMC2: Video Based Point Cloud Compression VPCC*. [Online]. Available: https://github.com/MPEGGroup/mpeg-pcc-tmc2

[5] S. Schwarz, G. Martin-Cocher, D. Flynn, and M. Budagavi, *Common Test Conditions for Point Cloud Compression*, document ISO/IEC JTC1/SC29/WG11 w17766, Ljubljana, Slovenia, 2018.

[6] D. Liu, Y. Li, J. Lin, H. Li, and F. Wu, "Deep learning-based video coding: A review and a case study," *ACM Comput. Surv.*, vol. 53, no. 1, pp. 1–35, 2020.

[7] M. Quach, G. Valenzise, and F. Dufaux, "Learning convolutional transforms for lossy point cloud geometry compression," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2019, pp. 4320–4324.

[8] J. Wang, H. Zhu, H. Liu, and Z. Ma, "Lossy point cloud geometry compression via end-to-end learning," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 12, pp. 4909–4923, Dec. 2021.

[9] A. F. R. Guarda, N. M. M. Rodrigues, and F. Pereira, "Adaptive deep learning-based point cloud geometry coding," *IEEE J. Sel. Topics Signal Process.*, vol. 15, no. 2, pp. 415–430, Feb. 2021.

[10] M. Quach, G. Valenzise, and F. Dufaux, "Improved deep point cloud geometry compression," in *Proc. IEEE 22nd Int. Workshop Multimedia Signal Process. (MMSP)*, Sep. 2020, pp. 1–6.

[11] S. Biswas, J. Liu, K. Wong, S. Wang, and R. Urtasun, "Muscle: Multi sweep compression of LiDAR using deep entropy models," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 22170–22181.

[12] L. Huang, S. Wang, K. Wong, J. Liu, and R. Urtasun, "OctSqueeze: Octree-structured entropy model for LiDAR compression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1310–1320.

[13] Z. Que, G. Lu, and D. Xu, "VoxelContext-Net: An octree based framework for point cloud compression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 6038–6047.

[14] C. Fu, G. Li, R. Song, W. Gao, and S. Liu, "Octattention: Octree-based large-scale contexts model for point cloud compression," 2022, *arXiv:2202.06028*.

[15] L. Gao, T. Fan, J. Wan, Y. Xu, J. Sun, and Z. Ma, "Point cloud geometry compression via neural graph sampling," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2021, pp. 3373–3377.

[16] K. You and P. Gao, "Patch-based deep autoencoder for point cloud geometry compression," in *Proc. ACM Multimedia Asia*, Dec. 2021, pp. 1–7.

[17] J. Wang, D. Ding, Z. Li, and Z. Ma, "Multiscale point cloud geometry compression," in *Proc. Data Compress. Conf. (DCC)*, Mar. 2021, pp. 73–82.

[18] J. Wang, D. Ding, Z. Li, X. Feng, C. Cao, and Z. Ma, "Sparse tensor-based multiscale representation for point cloud geometry compression," 2021, *arXiv:2111.10633*.

[19] R. Mekuria, K. Blom, and P. Cesar, "Design, implementation, and evaluation of a point cloud codec for tele-immersive video," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 4, pp. 828–842, Apr. 2017.

[20] D. C. Garcia and R. L. de Queiroz, "Intra-frame context-based octree coding for point-cloud geometry," in *Proc. 25th IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2018, pp. 1807–1811.

[21] M. Krivokuca, P. A. Chou, and M. Koroteev, "A volumetric approach to point cloud compression—Part II: Geometry compression," *IEEE Trans. Image Process.*, vol. 29, pp. 2217–2229, 2020.

[22] D. Thanou, P. A. Chou, and P. Frossard, "Graph-based compression of dynamic 3D point cloud sequences," *IEEE Trans. Image Process.*, vol. 25, no. 4, pp. 1765–1778, Apr. 2016.

[23] C. Cao, M. Preda, V. Zakharchenko, E. S. Jang, and T. Zaharia, "Compression of sparse and dense dynamic point clouds—Methods and standards," *Proc. IEEE*, vol. 109, no. 9, pp. 1537–1558, Sep. 2021.

[24] L. Li, Z. Li, V. Zakharchenko, J. Chen, and H. Li, "Advanced 3D motion prediction for video-based dynamic point cloud compression," *IEEE Trans. Image Process.*, vol. 29, pp. 289–302, 2020.

[25] D. C. Garcia, T. A. Fonseca, R. U. Ferreira, and R. L. de Queiroz, "Geometry coding for dynamic voxelized point clouds using octrees and multiple contexts," *IEEE Trans. Image Process.*, vol. 29, pp. 313–322, 2020.

[26] P. Gomes, "Graph-based network for dynamic point cloud prediction," in *Proc. 12th ACM Multimedia Syst. Conf.*, Jun. 2021, pp. 393–397.

[27] R. L. de Queiroz and P. A. Chou, "Motion-compensated compression of dynamic voxelized point clouds," *IEEE Trans. Image Process.*, vol. 26, no. 8, pp. 3886–3895, Aug. 2017.

[28] C. Cai, L. Chen, X. Zhang, and Z. Gao, "End-to-end optimized ROI image compression," *IEEE Trans. Image Process.*, vol. 29, pp. 3442–3457, 2020.

[29] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," 2018, *arXiv:1802.01436*.

[30] A. Djelouah, J. Campos, S. Schaub-Meyer, and C. Schroers, "Neural inter-frame compression for video coding," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6420–6428.

[31] G. Lu, W. Ouyang, D. Xu, X. Zhang, C. Cai, and Z. Gao, "DVC: An end-to-end deep video compression framework," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 10998–11007.

[32] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 77–85.

[33] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.

[34] T. Huang and Y. Liu, "3D point cloud geometry compression on deep learning," in *Proc. 27th ACM Int. Conf. Multimedia*, Oct. 2019, pp. 890–898.

[35] D. T. Nguyen and A. Kaup, "Learning-based lossless point cloud geometry coding using sparse tensors," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2022, pp. 2341–2345.

[36] C. Choy, J. Gwak, and S. Savarese, "4D spatio-temporal ConvNets: Minkowski convolutional neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3070–3079.

[37] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, Inception-ResNet and the impact of residual connections on learning," in *Proc. AAAI Conf. Artif. Intell.*, 2017, vol. 31, no. 1.

[38] E. d'Eon, B. Harrison, T. Myers, and P. A. Chou, *8I Voxelized Full Bodies—A Voxelized Point Cloud Dataset*, document ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) Input Document WG11M40059/WG1M74006, 2017.

[39] Y. Xu, Y. Lu, and Z. Wen, *Owlii Dynamic Human Mesh Sequence Dataset*, document ISO/IEC JTC1/SC29/WG11 m41658, 2017.

[40] D. Tian, H. Ochimizu, C. Feng, R. Cohen, and A. Vetro, "Geometric distortion metrics for point cloud compression," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 3460–3464.

[41] G. Bjontegaard, *Calculation of Average PSNR Differences Between RD-Curves*, document VCEG-M33, 2001.

Anique Akhtar (Member, IEEE) received the B.S. degree in electrical engineering from the Lahore University of Management Sciences (LUMS), Lahore, Pakistan, in 2013, the M.S. degree from Koc University, İstanbul, Turkey, in 2015, and the Ph.D. degree from the University of Missouri-Kansas City (UMKC), USA, in 2022. He was with the Wireless Communication Laboratory and the Multimedia and Communication Laboratory in the past. He is currently a Senior Engineer with the Multimedia R&D & Standards Group, Qualcomm Technologies Inc., San Diego, CA, USA, where he actively participates and contributes to the standardization efforts on MPEG's point cloud compression (PCC) and video-based dynamic mesh coding (V-DMC). His research interests include immersive video, point cloud and mesh compression, XR, and deep learning solutions for 3D data compression.

Zhu Li (Senior Member, IEEE) received the Ph.D. degree in electrical and computer engineering from Northwestern University in 2004. He was an AFRL Summer Faculty Member of the UAV Research Center, U.S. Air Force Academy (USAFA), from 2016 to 2018 and from 2020 to 2023. He was a Senior Staff Researcher with the Samsung Research America's Multimedia Standards Research Laboratory, Richardson, TX, USA, from 2012 to 2015, a Senior Staff Researcher with the Futurewei Technology's Media Laboratory, Bridgewater, NJ, USA, from 2010 to 2012, an 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 Laboratories, from 2000 to 2008. He is currently a Professor with the Department of Computer Science and Electrical Engineering, University of Missouri-Kansas City, where he is also the Director of the NSF I/UCRC Center for Big Learning (CBL). His research interests include point cloud and light field compression, graph signal processing and deep learning in the next gen visual compression, and image processing and understanding. He has more than 50 issued or pending patents, more than 190 publications in book chapters, journals, and conferences in these areas. He received the Best Paper Award at IEEE International Conference on Multimedia & Expo (ICME), Toronto, in 2006, and IEEE International Conference on Image Processing (ICIP), San Antonio, in 2007. He is also the Associate Editor-in-Chief of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY and an Associate Editor of IEEE TRANSACTIONS ON IMAGE PROCESSING (2020), IEEE TRANSACTIONS ON MULTIMEDIA (2015–2018), and IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY (2016–2019).

Geert Van der Auwera (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from Arizona State University, Tempe, AZ, USA, in 2007, and the M.S.E.E. (Belgian) degree from Vrije Universiteit Brussel (VUB), Brussels, Belgium, in 1997. He is currently the Director of the Multimedia R&D & Standards Group, Qualcomm Technologies Inc., San Diego, CA, USA, where he actively contributes to the standardization efforts on MPEG's dynamic mesh compression, point cloud compression, and previously on JVET's versatile video coding (VVC). Until January 2011, he was with Samsung Electronics, Irvine, CA, USA. Until December 2004, he was a Scientific Advisor with IWT-Flanders, Institute for the Promotion of Innovation by Science and Technology, Flanders, Belgium. In 2000, he joined IWT-Flanders after researching wavelet video coding with IMEC, Electronics and Information Processing Department (VUB-ETRO), Brussels, Belgium. His M.S.E.E. thesis on motion estimation in the wavelet domain received the Barco and IBM prizes by the Fund for Scientific Research of Flanders, Belgium, in 1998. His research interests are point cloud compression, XR, video coding, video traffic and quality characterization, video streaming mechanisms, and protocols.