Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000. Digital Object Identifier 10.1109/ACCESS.2017.DOI Range Image-Based Implicit Neural Compression for LiDAR Point Clouds

Kuwabara, Akihiro; Kato, Sorachi; Koike-Akino, Toshiaki; Fujihashi, Takuya

TR2026-023 February 19, 2026

Abstract

This paper presents a novel scheme to efficiently compress Light Detection and Ranging (LiDAR) point clouds, enabling high-precision 3D scene archives, and such archives pave the way for a detailed understanding of the corresponding 3D scenes. We focus on 2D range images (RIs) as a lightweight format for representing 3D LiDAR observations. Although conventional image compression techniques can be adapted to improve compression efficiency for RIs, their practical performance is expected to be limited due to differences in bit precision and the distinct pixel value distribution characteristics between natural images and RIs. We propose a novel implicit neural representation (INR)–based RI compression method that effectively handles floating-point valued pixels. The proposed method divides RIs into depth and mask images and compresses them using patch-wise and pixel-wise INR architectures with model pruning and quantization, respectively. Experiments on the KITTI dataset show that the proposed method outperforms existing image, point cloud, RI, and INR-based compression methods in terms of 3D reconstruction and detection quality at low bitrates and decoding latency.

IEEE Access 2026

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Range Image-Based Implicit Neural Compression for LiDAR Point Clouds

AKIHIRO KUWABARA[1], (Non Member, IEEE), SORACHI KATO[1], TOSHIAKI KOIKE-AKINO[2], and TAKUYA FUJIHASHI[1], (Member, IEEE)

[1]Graduate School of Information Science and Technology, The University of Osaka, Japan (e-mail: kuwabara.akihiro@ist.osaka-u.ac.jp)
[2]Mitsubishi Electric Research Laboratories (MERL), 201 Broadway, Cambridge, MA 02139, USA

CORRESPONDING AUTHOR: AKIHIRO KUWABARA (e-mail: kuwabara.akihiro@ist.osaka-u.ac.jp).

This work was supported by JST·ASPIRE Grant Number JPMJAP2432.

ABSTRACT This paper presents a novel scheme to efficiently compress Light Detection and Ranging (LiDAR) point clouds, enabling high-precision 3D scene archives, and such archives pave the way for a detailed understanding of the corresponding 3D scenes. We focus on 2D range images (RIs) as a lightweight format for representing 3D LiDAR observations. Although conventional image compression techniques can be adapted to improve compression efficiency for RIs, their practical performance is expected to be limited due to differences in bit precision and the distinct pixel value distribution characteristics between natural images and RIs. We propose a novel implicit neural representation (INR)–based RI compression method that effectively handles floating-point valued pixels. The proposed method divides RIs into depth and mask images and compresses them using patch-wise and pixel-wise INR architectures with model pruning and quantization, respectively. Experiments on the KITTI dataset show that the proposed method outperforms existing image, point cloud, RI, and INR-based compression methods in terms of 3D reconstruction and detection quality at low bitrates and decoding latency.

INDEX TERMS LiDAR, Point Clouds, Range Image INR.

I. INTRODUCTION

LIDAR sensors have gained significant attention not only in online applications but also in offline applications. In such offline applications, memory-efficient and precise three-dimensional (3D) scenes should be stored in advance and the 3D scenes should be smoothly retrieved from the storage based on the user demand for applications of 3D scene understanding such as digital archiving, environmental monitoring, navigation, and geological surveying [1]–[3]. LiDAR sensors scan the physical space with the ego-centric coordinate and measure the distance to the closest point on surrounding objects for each angle, allowing the creation of a point cloud with 3D points corresponding to the intersection of laser beams with objects ahead. As the resolution of LiDAR sensors increases, effectively storing and transmitting LiDAR scans becomes a significant challenge, primarily due to the substantial volume of each LiDAR sequence.

Although LiDAR scans are typically represented as 3D point clouds, they can also be expressed as a single-channel image, referred to as a two-dimensional (2D) range image (RI) [4], where point clouds captured in an ego-centric spherical coordinate system are projected onto a panoramic image. The x-y axes of the RI image correspond to the azimuth and elevation angles in the 3D spherical coordinate system, while each pixel value represents the distance to the corresponding point in that direction. Whereas 3D point clouds require $3N$ values to represent the locations of N point measurements, RIs require only N pixels at most, thus demonstrating their compactness.

We can pursue methods to further compress RIs. One potential solution is to adapt conventional lossy image compression techniques, such as Joint Photographic Experts Group (JPEG) [5] and Joint Photographic Experts Group 2000 (JPEG2000) [6]. However, these methods are based on integer precision for pixel values, which is incompatible for RIs whose pixels are represented by single or double precision floating-point numbers to precisely express the distance to the corresponding point measurements. We can still utilize these compression methods by adjusting the bit precision of RIs to align with them, but this naturally leads to degradation in 3D point cloud reconstruction performance due to inadequate distance resolution. Another drawback of

conventional compression methods lies in their strategy: they use block-based discrete cosine transform (DCT) and apply coarse quantization to high-frequency components based on human visual perception. This approach leads to significant degradation in the decoding performance of RIs, as RIs are characterized by large and sudden changes in the pixel value between foreground objects and distant backgrounds or pixels without any assigned point measurement.

To effectively compress RIs while preserving their fine precision and high-frequency changes in pixel value, this paper presents a novel RI compression method inspired by implicit neural representation (INR)-based image compression technique [7]. INR [8], [9] is a lightweight representation of multidimensional signals by compressing them into shallow neural networks (NNs). Specifically, INR overfits NNs with a limited number of parameters to the signals of interest through supervised learning, and the trained parameters become the compressed signal representation by providing the mapping function from signal indices, for example, coordinates on the image plane, to the corresponding signals. A primary challenge for INR-based signal compression is to ensure the precision of high-frequency details while simultaneously managing the constraints imposed by the limited model size. To address this challenge, we propose an extended INR training approach that incorporates both a mask image and an RI for point depth information. The mask image is a binary map that indicates whether each pixel corresponds to a projected 3D point (one) or not (zero). We begin by generating a mask image from the RI, followed by learning two separate coordinate-to-value mappings using distinct INR architectures: one for depth INR and one for the mask INR. During decoding, these trained INRs reconstruct both the depth and mask images, and the final reconstructed RI is obtained by applying the mask to the depth image. Although the reconstructed depth image may contain values for pixels that do not correspond to any 3D points, applying the mask enforces hard thresholding, effectively removing these artifacts. This process ensures a high-quality reconstructed RI, with sharp edges accurately preserved.

The contributions of our study are three-fold:

- To the best of our knowledge, this is the first paper to propose an INR-based intra RI compression method specifically designed for LiDAR measurements projected onto high-precision floating-point images.
- We extend the INR compression approach to incorporate the mask INR that explicitly represents whether any point measurements are assigned to each pixel on the RI or not, allowing efficient elimination of false point estimation on reconstructed depth images in the decoding process.
- We evaluate our proposed method using KITTI dataset [10] and compare its performance with existing baselines including conventional image compression, point cloud compression (PCC) [11], and RI-based and INR-based image compression, and show the better rate-distortion (R-D) performance and downstream task quality of our method than the baselines, especially at low bitrates.

II. RELATED WORK

A. POINT CLOUD COMPRESSION

The measured distance from LiDAR sensors is usually represented as a 3D point cloud. Each point cloud consists of a set of 3D points, and each point is defined by 3D coordinates, *i.e.*, (X, Y, Z). The graph-based and tree-based compression methods have been proposed to compress coordinate information, that is, geometry information. The graph-based methods regard the 3D points as graph signals and define graph Fourier transform (GFT) for frequency conversion in the graph domain. [12]–[14] utilized GFT for the geometry compression. Other studies [15], [16] reduce the storage and transmission costs for graph signal reconstruction. The tree-based compression method is another popular strategy for compressing geometry information. The typical way is octree-based representation, such as point cloud library (PCL) and geometry-based point cloud compression (G-PCC) [11], [17]. Some recent studies have been proposed to improve the efficiency of geometry compression using traditional signal processing [18] and deep neural network (DNN) [19], [20] solutions, respectively. For example, the study in [18] adaptively adopts the quad-tree (QT) and binary-tree (BT) block partitions in addition to those of octrees to improve the efficiency of the coding.

B. LIDAR RANGE IMAGE COMPRESSION

Many recent works consider projecting the measured LiDAR information onto 2D RI to represent the measured distance information in a compact format. There are two types of input LiDAR information to obtain the corresponding RIs: 1) raw packet containing the LiDAR laser IDs [21], the rotation angle of the LiDAR sensors and the distance values, and 2) 3D point clouds [22]–[24]. Our study utilizes 3D point clouds. The obtained RIs are then intra-coded [22] or inter-coded [23], [24] in lossless and lossy manners. Here, intra-coding reduces the spatial redundancy in each RI, whereas inter-coding reduces the temporal redundancy across RIs. In R-PCC [22], which is an intra-coding method, each RI can be coded by using a lossless coding method, such as LZ4 and Deflate, to compress the floating-point format. Our study is designed for RI intra-coding and exploits the INR-based compression to represent LiDAR measurements in small storage and transmission costs.

C. IMPLICIT NEURAL COMPRESSION

Since the concept of INR overfits multi-dimensional signals to a small NN architecture, recent studies exploit INR architectures for image compression. Specifically, each INR architecture takes a spatial/time index of the target signals and/or the corresponding feature vector to reconstruct the corresponding attribute values, such as color information. The overfitted weights of the INR architecture are shared with the receiver's side for signal reconstruction.

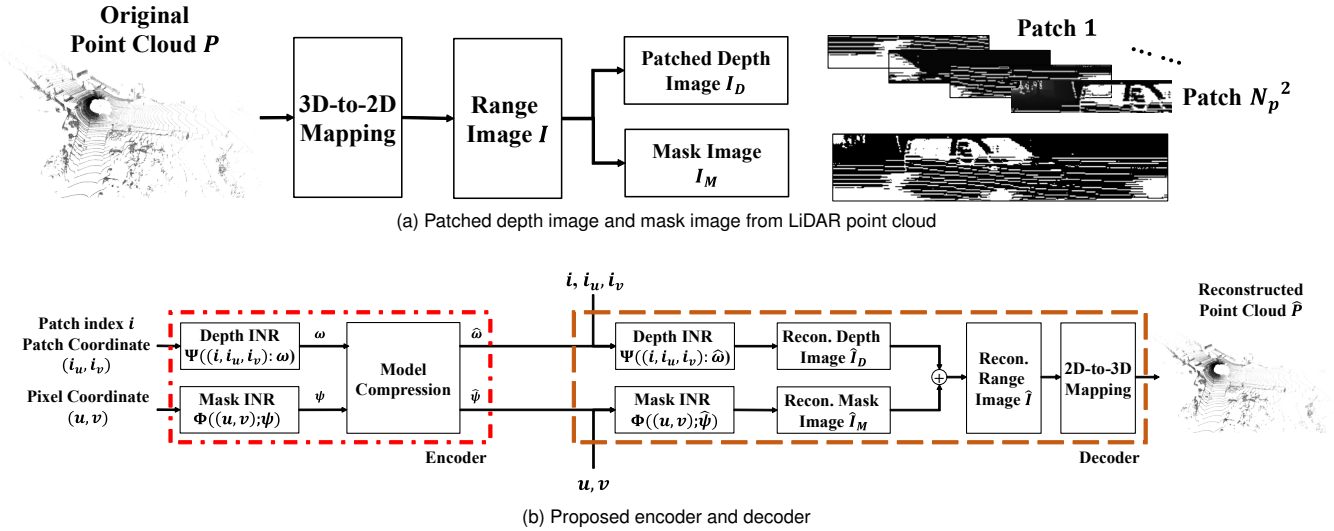


FIGURE 1: Overview of the proposed scheme.

The existing INR-based compression can be classified into pixel-wise, patch-wise, and frame-wise architectures. The frame-wise architectures realized inter-coding between multiple video frames to remove temporal redundancy. They feed the frame index and/or the corresponding embeddings to the NNs to generate each frame. Neural Representations for Videos (NeRV) [25] is the first work on frame-wise video compression, and various extensions [26]–[35] are proposed to improve the quality of reconstruction. However, NeRV architectures are large models when used for intra-coding each image.

The pixel-wise INR architecture [7], [36] takes the pixel index as input and reconstructs the corresponding pixel value. The patch-wise INR architecture was first proposed in [37]. Specifically, each image is divided into multiple patches, and the INR architecture takes the patch index as input to exploit the similarity of local adjacent pixels for high-quality reconstruction under the same model size. A key issue in such INR architectures is the lack of precision in high-frequency details with a small NN architecture. To represent high-frequency details under a small NN architecture, SIREN in [8] argues that sinusoidal activations work better than Rectified Linear Unit (ReLU) networks because sinusoidal activations can fit signals contained in higher-order derivatives. To address the same problem, our paper extends the training process of INRs by separating RIs into depth and mask images.

III. PROPOSED SCHEME

A. OVERVIEW

Fig. 1 shows an end-to-end architecture of the proposed scheme. Fig. 1 (a) specifically shows the procedure to obtain RI and corresponding depth and mask images from the LiDAR 3D point cloud. We consider that the LiDAR measurement to be compressed is a 3D point cloud consisting of N points, denoted as $\mathbf{P} = \{\mathbf{p}_i = [x_i, y_i, z_i] \mid i = 1, \cdots, N\}$, where $x_i, y_i, z_i \in \mathbb{R}$ represent the Cartesian coordinates of the i-th point. The point cloud is first transformed into the spherical coordinate system. Subsequently, each point is projected onto a 2D image plane by mapping it to a pixel in a single-channel image $I \in \mathbb{R}^{W \times H}$, producing an RI. Here, W and H represent the width and height of the range image, respectively. The RI is then divided into a depth image $I_D \in \mathbb{R}^{W \times H}$ and a mask image $I_M \in \{0,1\}^{W \times H}$. The depth image I_D is further segmented into small rectangular regions, or patches, with a resolution of $\frac{W}{N_p} \times \frac{H}{N_p}$, where N_p is the scaling factor.

Fig. 1 (b) shows the sequential operations of the encoder and decoder. In the encoder, two distinct INRs, namely the mask INR $\Phi(\cdot;\boldsymbol{\psi})$ and the depth INR $\Psi(\cdot;\boldsymbol{\omega})$ with learnable parameters $\boldsymbol{\psi}$ and $\boldsymbol{\omega}$, are trained to be overfitted to the depth image and the mask image, respectively. This training process is a pixel-wise process, which means that the parameters are trained to obtain a mapping from the pixel coordinates or patch indices on the images to their corresponding pixel values. This is achieved by sequentially providing a pair of indices to the networks. The well-trained parameters $\boldsymbol{\psi}$ and $\boldsymbol{\omega}$ are subsequently pruned and quantized as $\hat{\boldsymbol{\psi}}$ and $\hat{\boldsymbol{\omega}}$ to enhance their compactness, and we assume that these parameters are stored in storage or transmitted to content receivers as the lightweight format of the LiDAR measurements. In the decoding process, the decoder uses compressed parameters to reconstruct a mask image $\hat{I}_M$ and a depth image $\hat{I}_D$ from individual INR architectures $\Phi(\cdot;\hat{\boldsymbol{\psi}})$ and $\Psi(\cdot;\hat{\boldsymbol{\omega}})$, respectively. Similarly to the encoding process, the images are reconstructed by sequentially feeding a pair of coordinates and indices to the INRs and collecting all estimated values to form the shape of the image. We obtain the final result of RI, $\hat{I}$, by applying the mask image to the depth image to mask out any values of the pixels in the depth image corresponding to the pixels with a mask value

of 0, indicating "no point". Finally, the LiDAR point cloud is reconstructed as $\hat{\mathbf{P}}$ from $\hat{I}$ via a reverse coordinate projection process from the 2D image plane, through the spherical coordinate system, to the Cartesian coordinate system.

B. 3D-TO-2D MAPPING

Our proposed method first performs a coordinate transformation for all points in the 3D point cloud $\mathbf{P}$ measured by LiDAR sensors to obtain a 2D RI I. Specifically, the 3D-to-2D mapping consists of two steps: 1) mapping points in the 3D Cartesian coordinate system x-y-z to the spherical coordinate ρ-ϕ-θ, and 2) mapping points in the spherical coordinate ρ-ϕ-θ to an image coordinate system u-v.

Each 3D point in the point cloud $\mathbf{p} \in \mathbf{P}$ consists of the 3D Cartesian coordinate (x, y, z) first. This point is transformed into a point in the spherical coordinate $\mathbf{p}' = (\rho, \phi, \theta)$, where ρ, ϕ, θ denotes the length, pitch, and yaw of the coordinate system, as follows:

$$\rho = \sqrt{x^2 + y^2 + z^2},\ \phi = \arcsin\left(\frac{z}{\rho}\right),\ \theta = \arctan\left(\frac{y}{x}\right). \tag{1}$$

The point in the spherical coordinate is further transformed to the image coordinate (u, v) to generate 2D RI I as follows:

$$\begin{aligned} u &= \left\lfloor \frac{W}{2} \times \left(\frac{\theta}{\pi} + 1\right) \right\rfloor, \\ v &= \left\lfloor H \times \left(1 - \frac{\phi + |\phi_{\text{down}}|}{\phi_{\text{up}} + |\phi_{\text{down}}|}\right) \right\rfloor, \end{aligned} \tag{2}$$

where ϕ_{up} and ϕ_{down} are the maximum and minimum value of ϕ in the dataset, $|\cdot|$ is the absolute value, and $\lfloor \cdot \rfloor$ is a floor function. H and W in Eq. (2) are the height and width of RI, and they are determined by the angular resolution of the LiDAR sensor for the elevation and azimuth axes. In this study, we set $H = 64$ and $W = 1024$. The value of each pixel $I(u, v)$ on the RI is the measured distance ρ, derived in Eq. (1), with arbitrary unit for physical length.

Due to the sparsity of LiDAR measurements, not all pixels on the RI are guaranteed to be assigned to any 3D point. Therefore, if a pixel on (u', v') remains unassigned after performing the 3D-to-2D mapping for all points, we set $I(u', v') = \rho_{null}$ where ρ_{null} is the arbitrary value indicating that no 3D point is assigned to the pixel. In practice, ρ_{null} should be selected to be greater than the maximum value of ρ in the LiDAR measurements or a negative value.

C. DEPTH/MASK IMAGE CONSTRUCTION

After 3D-to-2D mapping, the RI is then divided into a mask image $I_M \in \{0, 1\}^{W \times H}$ and a depth image $I_D \in \mathbb{R}^{W \times H}$.

The mask image I_M is to indicate whether any 3D point is assigned to each pixel on the RI or not and is defined as:

$$I_M(u, v) = \begin{cases} 1 & \text{if } I(u, v) \neq \rho_{null}, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Given the mask image, we construct a dataset $\mathcal{D}_M$ for training the mask INR $\Phi(\cdot; \boldsymbol{\psi})$ which consists of pairs of the coordinates of pixels and corresponding binary values as

$$\mathcal{D}_M = \{((u, v), I_M(u, v)) \mid u \in \{1, \ldots, W\}, v \in \{1, \ldots, H\}\}. \tag{4}$$

The depth image I_D is the masked version of the RI. Pixels without any 3D point assignment are considered as "Do not care" ($\emptyset$) and are defined as such.

$$I_D(u, v) = \begin{cases} \emptyset & \text{if } I(u, v) = \rho_{null}, \\ I(u, v) & \text{otherwise.} \end{cases} \tag{5}$$

In addition, we divide the depth image into small rectangular areas, or patches, inspired by recent works [37] to improve the decoding performance and the quality of the reconstructed depth image. Specifically, the RI is evenly segmented into patches $I'_D(i) \in \mathbb{R}^{\frac{W}{N_p} \times \frac{H}{N_p}}$, where N_p is a scaling factor and $i = 1, \cdots, N_p^2$. Each patch is assigned a patch index, allowing us to specify a pixel in the patched RI as $I'_D(i, i_u, i_v)$, where i represents the patch index and i_u, i_v denotes the in-patch pixel coordinates whose origin is the top left pixel in the i-th patch. Similarly to the mask image, we also construct a dataset $\mathcal{D}_D$ for training depth INR $\Psi(\cdot; \boldsymbol{\omega})$ which consists of pairs of a patch index, in-patch coordinates, and the corresponding depth values, excluding unassigned pixels, as follows:

$$\begin{aligned} \mathcal{D}_D = \{((i, i_u, i_v), I_D(i, i_u, i_v)) \mid\ & i \in \{1, \ldots, N_p^2\}, \\ & i_u \in \{1, \ldots, W/N_p\}, \\ & i_v \in \{1, \ldots, H/N_p\}, \\ & I_D(i, i_u, i_v) \neq \emptyset\} \end{aligned} \tag{6}$$

D. INR-BASED RI ENCODER

In the encoding process, the mask INR $\Phi(\cdot; \psi)$ and the depth INR $\Psi(\cdot; \omega)$ are trained to obtain good parameters to express the coordinate-to-value relationships contained in the mask dataset $\mathcal{D}_M$ and $\mathcal{D}_D$. Figs. 2 (a) and (b) show the detailed architecture of the proposed depth INR and mask INR, respectively.

1) Mask INR

Regarding the mask image, we assume the existence of a function Φ_M, which maps each coordinate on the image to a binary value as

$$\Phi_M : \mathbb{R}^2 \longrightarrow \{0, 1\}, \tag{7}$$

and the objective of the mask INR is to obtain parameters $\boldsymbol{\psi}$ that well approximate that mapping function as $\Phi(\cdot; \boldsymbol{\psi}) \approx \Phi_M$ through supervised learning with the dataset $\mathcal{D}_M$. Fig. 2 (a) shows the detailed architecture of the proposed mask INR. The mask INR is a multi-layer perceptron (MLP) with L hidden layers and V nodes, and after each hidden layer, a sinusoidal function layer is used as the activation function. The network sequentially receives the coordinate (u, v) from the mask dataset $\mathcal{D}_M$ and regresses

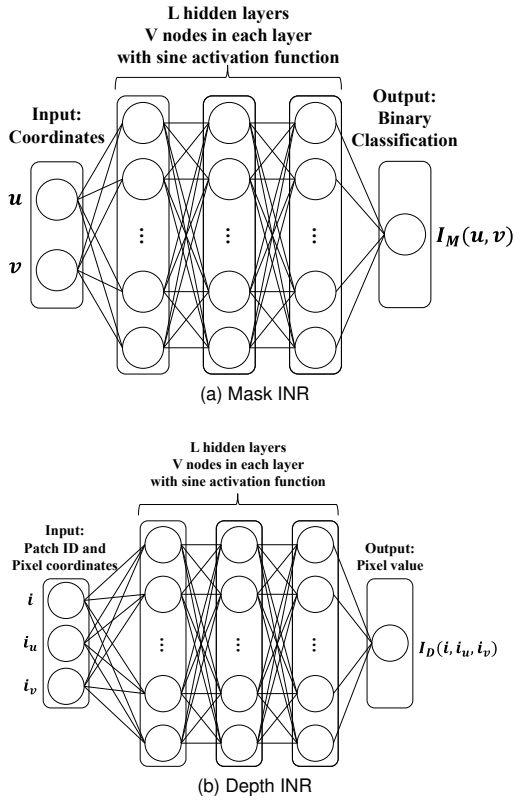


FIGURE 2: Architectures of the proposed mask and depth INRs.

a binary value as its output. Regression loss is computed using binary cross entropy (BCE) loss function between all output values $\Phi((u,v);\boldsymbol{\psi})$ and the corresponding true values $I_M(u,v)$ as follows:

$$\mathcal{L}_{\mathrm{BCE}}(\boldsymbol{\psi}) = -\frac{1}{HW}\sum_{u}^{H}\sum_{v}^{W}[I_M(u,v)\log\left(\Phi((u,v);\boldsymbol{\psi})\right) + (1-I_M(u,v))\log\left(1-\Phi((u,v);\boldsymbol{\psi})\right)]. \tag{8}$$

2) Depth INR

Similar to the mask INR, we also assume the existence of another function Ψ_D regarding the depth image, which maps the pair of a patch index and an in-patch pixel coordinate to a depth value as

$$\Psi_D : \mathbb{R}^3 \longrightarrow \mathbb{R}^1. \tag{9}$$

and the objective of the depth INR is to obtain parameters $\boldsymbol{\omega}$ for good approximation of Ψ_D as $\Psi((i,i_u,i_v);\boldsymbol{\omega}) \approx \Psi_D$. Figs. 2 (b) shows the detailed architecture of the proposed depth INR. The depth INR shares its structure with the mask INR, with the exception of the input layer, which accepts 3 values. In the training process, pairs of patch index i and an in-patch coordinate (i_u,i_v) are sequentially passed to the depth INR network from the depth dataset $\mathcal{D}_D$, and the corresponding depth values are regressed. We employ mean squared error (MSE) loss as a regression loss function for the depth INR as

$$\mathcal{L}_{\mathrm{MSE}}(\boldsymbol{\omega}) = \frac{1}{HW}\sum_{i}^{N_p^2}\sum_{i_u}^{W/N_p}\sum_{i_v}^{H/N_p}\|\Psi((i,i_u,i_v);\boldsymbol{\omega}) - I_D(i,i_u,i_v)\|^2. \tag{10}$$

E. MODEL COMPRESSION

Parameters $\boldsymbol{\psi}$ and $\boldsymbol{\omega}$ become effectively compressed representations of the depth and mask images after a thorough training. We introduce a series of parameter compression processes for both to further improve their compactness.

1) Model Pruning

As an initial step in our parameter compression procedure, we implement global unstructured pruning for parameters in both depth and mask INRs. Given a threshold w_q for the magnitude of parameters, each parameter w is determined to be retained or pruned based on the following criteria:

$$\hat{\boldsymbol{\omega}} = \begin{cases} \boldsymbol{\omega} & \boldsymbol{\omega} \geq \boldsymbol{\omega}_q, \\ \mathbf{0} & \text{otherwise.} \end{cases} \tag{11}$$

To guarantee that the pruned parameters are of good expression, we subsequently retrain the parameters to fine-tune using the same dataset $\mathcal{D}_M$ and $\mathcal{D}_D$.

2) Model Quantization and Encoding

The pruned and fine-tuned parameters are uniformly quantized to a bit depth of N_b. This quantization is layer-wise, meaning that given a parameter set corresponding to each layer in the depth and mask INRs as $\boldsymbol{\mu} \in \hat{\boldsymbol{\omega}}$, a quantized parameter set $\boldsymbol{\mu}_q$ is obtained as follows:

$$\boldsymbol{\mu}_q = \mathrm{round}\left(\frac{\boldsymbol{\mu}-\boldsymbol{\mu}_{\min}}{2^{N_b}}\right)s + \boldsymbol{\mu}_{\min},\ s = \frac{\boldsymbol{\mu}_{\max}-\boldsymbol{\mu}_{\min}}{2^{N_b}}, \tag{12}$$

where $\mathrm{round}(\cdot)$ is a rounding function to the nearest integer and $\boldsymbol{\mu}_{\max}$ and $\boldsymbol{\mu}_{\min}$ are the maximum and minimum values in $\boldsymbol{\mu}$. The quantized tensor $\boldsymbol{\mu}_q$ is finally coded into a binary sequence using Huffman coding. It is noteworthy that the quantized parameters $\boldsymbol{\mu}_q$ are likely to assume values near zero, particularly for smaller bit depths. Consequently, Huffman coding demonstrates its effectiveness in reducing the overall size of encoded parameters.

F. RI DECODER

The decoding process of RI is a simple feedforward process involving the mask INR and depth INR with optimized parameters $\hat{\boldsymbol{\psi}}$ and $\hat{\boldsymbol{\omega}}$. The mask image is reconstructed by feeding the coordinate sets $\{(u,v) \mid u \in \{1,\cdots,W\}, v \in \{1,\cdots,H\}\}$ to the mask INR. The resulting binary values are then reshaped to construct the $W \times H$ mask image $\hat{I}_M$.

The depth image reconstruction is in a two-stage manner. Each patch is first reconstructed by feeding the sets of pairs of a patch index and an in-patch coordinate $\{(i,i_u,i_v) \mid i \in$

$\{1, \cdots, N_p^2\}, i_u \in \{1, \cdots, W/N_p\}, i_v \in \{1, \cdots, H/N_p\}\}$ to the depth INR. The reconstructed patches are then gathered to build the complete depth image $\hat{I}_D$. Finally, the reconstructed RI $\hat{I}$ is obtained as

$$\hat{I}(u,v) = \begin{cases} \hat{I}_D(u,v) & \hat{I}_M(u,v) = 1, \\ \rho_{\text{null}} & \text{otherwise.} \end{cases} \tag{13}$$

G. 2D-TO-3D MAPPING

The concluding phase of our decoding procedure is the reconstruction of a 3D point cloud through a 2D-to-3D mapping against the reconstructed RI. When the pixel (u, v) on the RI has a valid point depth, i.e., is not ρ_{null}, the corresponding point in the spherical coordinate $\hat{\mathbf{p}}' = (\hat{\rho}, \hat{\phi}, \hat{\theta})$ is obtained as follows:

$$\begin{aligned} \hat{\rho} &= \hat{I}(u,v), \\ \hat{\phi} &= \left(1 - \frac{v}{H}\right)(\phi_{up} + |\phi_{\text{down}}|) - |\phi_{\text{down}}|, \\ \hat{\theta} &= -\left(2\frac{u}{W} - 1\right)\pi. \end{aligned} \tag{14}$$

Finally, the 3D points in the spherical coordinate are transformed into the 3D Cartesian coordinate $\hat{\mathbf{p}} = (\hat{x}, \hat{y}, \hat{z})$ as

$$\hat{x} = \hat{\rho}\cos\hat{\phi}\cos\hat{\theta},\ \hat{y} = \hat{\rho}\cos\hat{\phi}\sin\hat{\theta}\ , \hat{z} = \hat{\rho}\sin\hat{\phi}. \tag{15}$$

IV. EXPERIMENTS

A. SETTINGS

Dataset: We use the KITTI dataset [10] as our source of 3D point cloud data. For R-D performance, we evaluate the KITTI Odometry dataset, using frames 00, 25, 50, 75, and 100 from sequences 00 to 06. For downstream tasks, we use the KITTI 3D Object Detection dataset. We split the official training set into 3,712 training samples and 3,769 validation samples, and evaluate detection performance on the validation split to quantify the impact of compression. We use OpenPCDet [38] v0.6.0 to train and evaluate three representative detectors: PointPillars [39], SECOND [40], and PointRCNN [41].

All data were collected with a Velodyne 64 scanner that features 64 laser scan lines and an azimuth resolution of 0.09 degrees. In the proposed scheme, these 3D point clouds are projected into an RI for compression. Note that projecting points into a range image may cause point loss. We evaluate the point retention ratio defined as $r_N = |\hat{\mathcal{P}}|/|\mathcal{P}|$, where $\mathcal{P}$ is the original input point cloud and $\hat{\mathcal{P}}$ is the reconstructed point cloud via RI projection and back-projection. On the KITTI odometry dataset, we observe $r_N = 41.02\% \pm 0.38\%$ using 35 selected frames from sequences 00–06. Similarly, we observe $r_N = 74.33\% \pm 0.69\%$ on the KITTI object detection validation split of 3,769 frames, measured on the cropped point clouds used as inputs for 3D object detectors.

Metric: Regarding the metrics for the decoded 3D point clouds, we follow the common practice in the community using chamfer distance (CD). CD has been widely adopted as a distortion measure for 3D point cloud reconstruction and sensing [42].

CD is defined as:

$$\text{CD} = \frac{1}{2}\left\{\frac{1}{|\mathbf{P}|}\sum_{\boldsymbol{p}\in\mathbf{P}}\min_{\hat{\boldsymbol{p}}\in\hat{\mathbf{P}}}\|\boldsymbol{p}-\hat{\boldsymbol{p}}\|_2 + \frac{1}{|\hat{\mathbf{P}}|}\sum_{\hat{\boldsymbol{p}}\in\hat{\mathbf{P}}}\min_{\boldsymbol{p}\in\mathbf{P}}\|\boldsymbol{p}-\hat{\boldsymbol{p}}\|_2\right\}, \tag{16}$$

where $\mathbf{P}$ is the set of 3D points in the original point cloud and $\hat{\mathbf{P}}$ is the set of 3D points in the decoded point set.

For the R-D performance assessment between the proposed method and the baselines, we use the Bjøntegaard delta chamfer distance (BD-CD) [43] for calculating average CD improvement between R-D curves for the same bitrate, where positive values denote CD improvement compared to the baselines. For downstream tasks, we evaluate 3D object detection accuracy using the Car 3D bounding-box average precision (AP).

Network Architecture Details: Both INR architectures are designed to effectively approximate the coordinate-to-value mappings in the mask and depth images derived from RI. The mask INR is an MLP with a fixed depth of $L = 6$ layers. We experimented with varying the number of nodes V in each hidden layer to evaluate the impact on performance and compression efficiency. The values of V considered are $\{10, 19, 24, 28, 31, 34, 37, 40\}$.

The network takes as input the pixel coordinates $(u, v) \in \mathbb{R}^2$ from the mask dataset $\mathcal{D}_M$ and outputs a scalar value representing the mask at that coordinate. The architecture is structured as follows:

- **Input Layer**: The coordinates of the pixels (u, v).
- **Hidden Layers**: It consists of $L = 6$ hidden layers, each with V nodes. Each hidden layer employs the sinusoidal activation function to introduce periodicity and enable the network to model high-frequency variations in the mask image.
- **Output Layer**: A single node with the sigmoid activation function produces an output in the range $(0, 1)$, suitable for binary classification of mask values.

The Depth INR is also implemented as an MLP with a fixed depth of $L = 6$ layers. We also consider the different number of nodes V in each hidden layer, choosing $\{28, 31, 34, 37, 40, 42, 45\}$ to evaluate the trade-off between model capacity and compression.

The input to the depth INR is a concatenation of the patch index i and the in-patch pixel coordinates $(i_u, i_v) \in \mathbb{R}^2$, resulting in a 3-dimensional input vector. Since we set the patch scaling factor to $N_p = 16$, the patch index i ranges from 0 to 255. The architecture of the depth INR is as follows:

- **Input Layer**: The concatenated input vector (i, i_u, i_v).
- **Hidden Layers**: It contains $L = 6$ hidden layers, each with V nodes. Similarly to the mask INR, the sinusoidal activation function is applied to each hidden layer to capture the complex variations in the depth image.
- **Output Layer**: A single node with a linear activation function (identity function) to output the estimated depth value $\hat{I}_D(i, i_u, i_v) \in \mathbb{R}$.

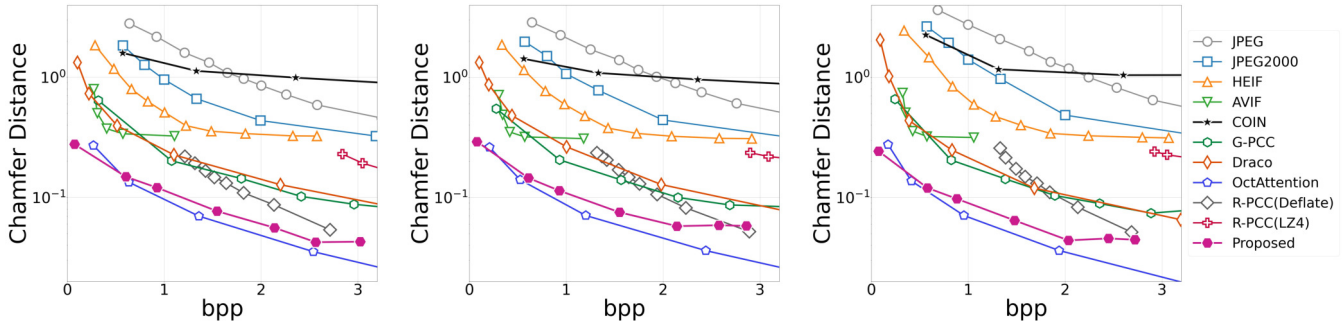


FIGURE 3: Chamfer distance as a function of bitrate across different sequences of the KITTI LiDAR point clouds, where the bitrate is measured in bits per point (bpp). From left to right: (left) performance in frame 00 of sequence 00, (middle) performance in frame 25 of sequence 00, and (right) performance in frame 50 of sequence 00.

Hyperparameter Details: We use separate hyperparameter settings for mask and depth INRs. The general settings for both INRs include the Adam optimizer, an initial learning rate of 1×10^{-3}, 3,000 training epochs, and a batch size of 1. For depth INR, we adopt the cosine annealing scheduler with a warmup phase. The initial learning rate is set to 1×10^{-4}, and the warmup period lasts for 300 epochs. The minimum learning rate is set to 1×10^{-12}.

Model Compression Details: A global unstructured pruning is used for model pruning. The pruning ratio (sparsity) was varied from 0 to 1 to adjust the sparsity of the model parameters. For each pruning ratio, we determined the corresponding threshold ω_q to control which parameters were pruned. A higher pruning ratio results in more parameters being set to zero. After pruning, we fine-tuned the model using the same dataset to recover any potential performance loss.

To further compress the pruned and fine-tuned model, we perform uniform quantization to the parameters. The quantization bit depth N_b was varied from 4 to 32 bits to balance compression performance and model precision. The quantized parameters were then encoded using Huffman coding to further reduce the model size.

Baselines: We evaluate our proposed method by comparing it with existing baselines in both geometric 3D point cloud compression and 2D image compression.

1) As baselines for 3D point cloud compression, we select **G-PCC** within the PCC family. We refer to the MPEG reference implementation TMC13-v14.0 for octree geometry compression.
2) We also select **Draco** [44] as the 3D point cloud compression baseline which also belongs to the PCC family. We use the official implementation of the Draco encoder that performs KD-tree-based compression [45].
3) **OctAttention** [19] is an octree-based autoencoder within the PCC family. This method improves the conventional octree structure by incorporating attention mechanisms for better context modeling. To evaluate its performance across different compression levels, we set the octree depth to values from 8 to 13.
4) As conventional image compression baselines, we select **JPEG**, **JPEG2000**, **High-Efficiency Image File Format (HEIF)**, and **AV1 Image File Format (AVIF)**. We convert floating-point valued RIs into 8-bit precision in advance when using these methods.
5) **R-PCC** [22] is an RI based LiDAR compression baseline. It maps LiDAR point clouds to RIs and performs intra-coding using floating-point lossless coding methods. Here, we use LZ4 and Deflate for coding methods due to their fast decompression.
6) **COmpression with Implicit Neural representations (COIN)** [7] is an INR–based image compression baseline. The INR architecture is trained to obtain a direct mapping of the pixel coordinate to each pixel value of RI. We assume that COIN serves as a reliable indicator of the efficiency of our depth/mask separation strategy, as it does not employ the process.

Implementation Detail: All the evaluations exhibited in this paper are performed with CPUs of Intel Core i9-10850K and i9-13900KF and with GPUs of NVIDIA GeForce RTX 3080 and 4070. NNs for COIN and our proposed method are implemented, trained, and evaluated using PyTorch 2.2.0 with Python 3.10.

B. COMPARISON WITH BASELINES

1) RATE-DISTORTION PERFORMANCE

We show the R-D performance of our proposed method and baselines. Figs. 3 show the CD between the original and reconstructed LiDAR point clouds against various bitrates, i.e., bit per point (bpp). We observe the following findings:

- The proposed method achieves higher 3D reconstruction quality than G-PCC, Draco, image compression, and RI compression methods across the bpp range up to 3.0 for frame 00.
- OctAttention achieves the best R-D performance in frames 00, 25 and 50, whereas it requires long decoding latency, as will be detailed in Table 2.
- Image compression methods suffer from quality saturation due to the precision disparity between RI and the

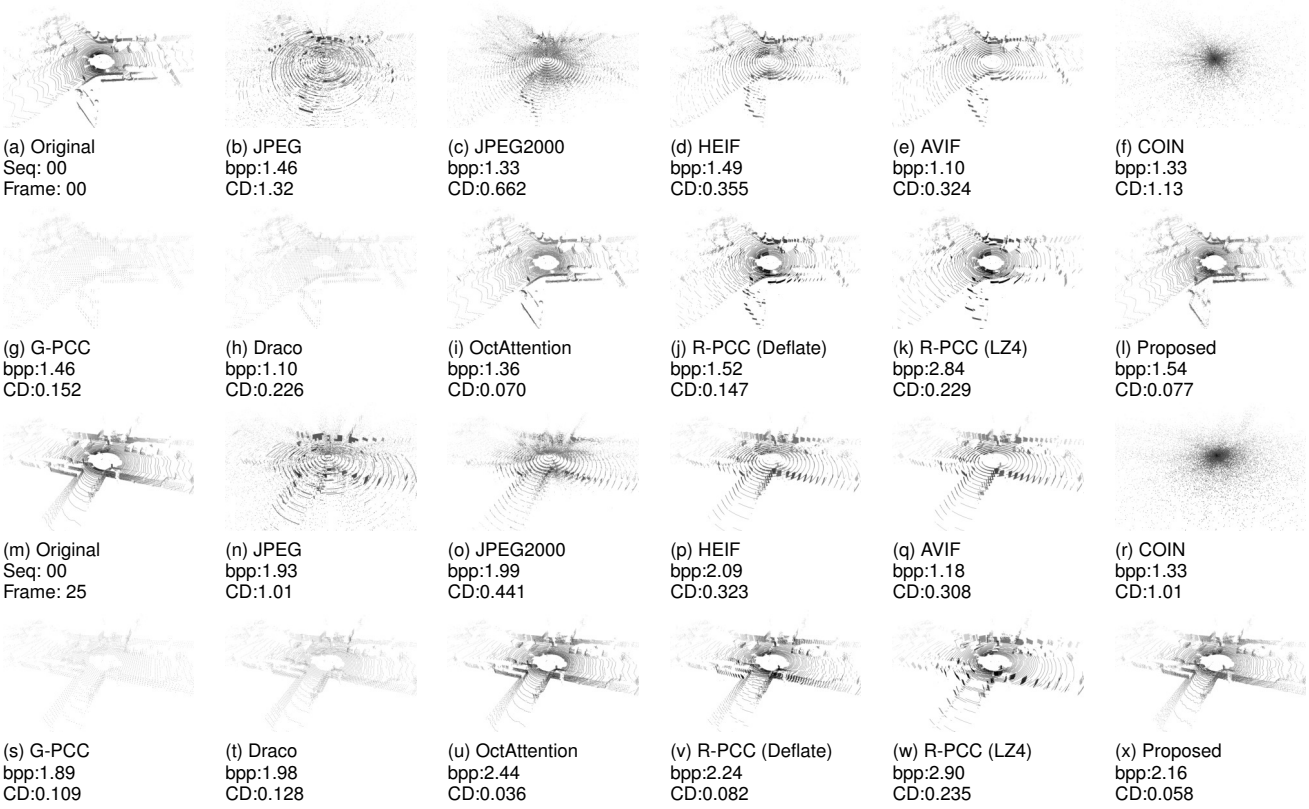


FIGURE 4: Snapshot of the reconstructed LiDAR point clouds in proposed and baseline methods. Here, (b)-(l) and (n)-(x) show the reconstructed point clouds of frames 00 and 25 of sequence 00, respectively.

TABLE 1: The list of BD-CD ↑ for the KITTI dataset across the different sequences. Note that BD-CD is evaluated for each baseline using the proposed method as the reference. Positive values indicate that the proposed method achieves a lower chamfer distance than the corresponding baseline.

Seq.	JPEG†	JPEG2000†	HEIF†	AVIF†	COIN‡	G-PCC§	Draco§	Oct Attention §	R-PCC (Deflate) ¶	R-PCC (LZ4) ¶
00	1.393	0.801	0.535	0.256	1.103	0.081	0.157	-0.025	0.023	0.110
01	1.234	0.686	0.515	0.259	1.095	0.097	0.234	-0.017	0.019	0.097
02	1.120	0.612	0.447	0.254	1.039	0.083	0.226	-0.026	0.006	0.094
03	1.322	0.733	0.497	0.232	1.074	0.059	0.176	-0.045	-0.007	0.078
04	1.343	0.748	0.515	0.237	1.092	0.074	0.192	-0.039	0.010	0.097
05	1.484	0.877	0.576	0.248	1.041	0.065	0.153	-0.040	0.008	0.095
06	1.324	0.785	0.527	0.224	0.993	0.053	0.153	-0.070	-0.024	0.062
Average	1.317	0.749	0.516	0.244	1.063	0.073	0.185	-0.037	0.005	0.090

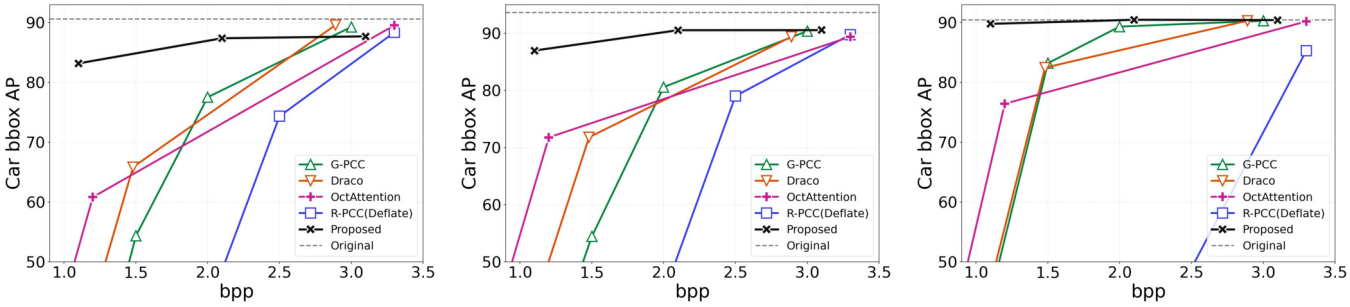
†: Image based method(s), ‡: INR based method, §: Point Cloud based method(s), ¶: Range Image based method(s).

typical 8-bit precision image.

- PCC methods do not have saturation since they compress the geometry information with 10-bit precision.
- In R-PCC, R-D performance highly depends on the lossless coding method.
- The INR-based method requires a large model size for reconstructing high-quality RI.

Figs. 4 (a)-(x) show the snapshots of the original and reconstructed LiDAR point clouds in each method Here, Figs. 4 (a)-(l) and (m)-(x) use frames 00 and 25 of sequence 00, respectively. The proposed method can reconstruct a clean point cloud at the same bitrate. However, some PCC, image compression, and INR-based compression methods contain circular noises and/or decrease the number of 3D points in the reconstructed LiDAR point clouds. A circular noise still remains in R-PCC methods as well.

Table 1 lists the average BD-CD performance of the proposed method against the baselines in each sequence of LiDAR point clouds. Here, BD-CD is evaluated for each baseline using the proposed method as the reference. It shows that the proposed method achieves the best 3D reconstruction quality in the same bitrate range against G-PCC, Draco, image compression, and INR-based compression methods irrespective of LiDAR sequences. For OctAttention and R-



(a) PointPillars (b) SECOND (c) PointRCNN

FIGURE 5: Quantitative results of 3D object detection on the KITTI detection dataset. Car 3D bounding-box AP@0.7, 0.7, 0.7 as a function of bitrate (bits per point), for three downstream detectors (left to right): PointPillars, SECOND, and PointRCNN.

TABLE 2: Average decoding latency ↓

Method	Latency per frame
JPEG	0.49 ms
JPEG2000	0.54 ms
HEIF	0.51 ms
AVIF	0.48 ms
COIN	0.71 ms
G-PCC	2.00 ms
Draco	3.00 ms
OctAttention	10.6 s
R-PCC (Deflate)	11.5 ms
R-PCC (LZ4)	10.3 ms
Proposed	0.69 ms

TABLE 3: Average encoding latency ↓

Method	Latency per frame
JPEG	9 ms
JPEG2000	10 ms
HEIF	55 ms
AVIF	94 ms
COIN	30 min
G-PCC	65 ms
Draco	10 ms
OctAttention	134 ms
R-PCC (Deflate)	20 ms
R-PCC (LZ4)	60 ms
Proposed	180 min

PCC (Deflate), the proposed method can be comparable or slightly worse in some cases. In addition, the performance gap between R-PCC and the proposed methods depends on the lossless coding method. When R-PCC uses a low-efficiency coding method, such as LZ4, for fast decoding, the proposed scheme achieves better R-D performance than R-PCC.

2) DOWNSTREAM TASK

We then discuss the impact of our RI compression method on the performance of downstream tasks on the LiDAR point cloud. We selected 3D object detection as a representative example of downstream tasks. To evaluate robustness across different perception architectures, we consider three representative LiDAR detectors: PointPillars (BEV-based), SECOND (voxel-based), and PointRCNN (point-based).

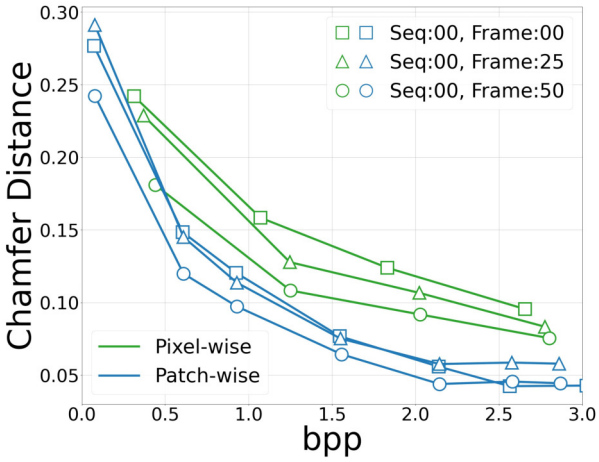
Fig. 5 shows the Car 3D bounding-box AP@0.7, 0.7, 0.7 as a function of bitrate for the original and reconstructed point clouds, evaluated with PointPillars, SECOND, and PointRCNN. The dashed line indicates the detection accuracy on the uncompressed point clouds. The results demonstrate that the proposed method achieves higher detection accuracy than the baselines across all three detectors in low-bpp regimes, i.e., bpp from 1.0 to 2.0.

3) DECODING LATENCY

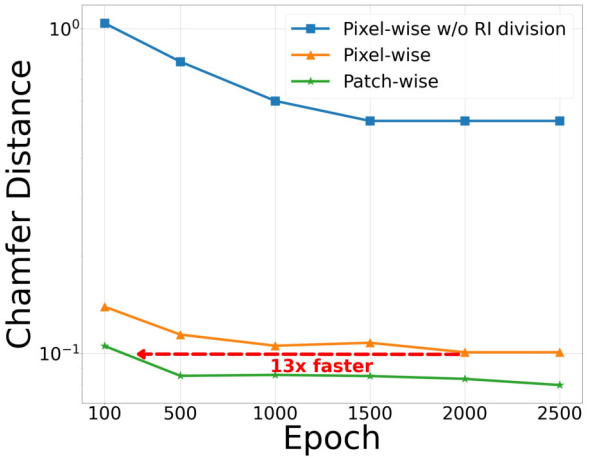
Table 2 shows the average decoding latency of the proposed and baseline methods for LiDAR frame 00 of sequence 00. The decoding latency values for the proposed method and the baselines are the total time required from RI decoding to the 2D-to-3D mapping. The decoding latency of the proposed method is comparable to that of image compression methods and has more than 65.5% and 93.3% reduction compared to G-PCC, Draco and R-PCC methods, respectively. In addition, the proposed scheme achieves a speedup of over four orders of magnitude compared to OctAttention. This means that the proposed method approaches the decoding latency of image compression methods and achieves 3D reconstruction quality comparable to PCC/R-PCC methods.

4) ENCODING LATENCY

Table 3 lists the average encoding latency of the proposed and baseline methods for LiDAR frame 00 of sequence 00. Here, the encoding latency for the RI-based schemes contains the conversion time from the point cloud to the RI. It can be seen that INR-based approaches, including the proposed one, involve significantly longer encoding time than both 3D point cloud and 2D image compression methods. However, as



(a) R-D performance.



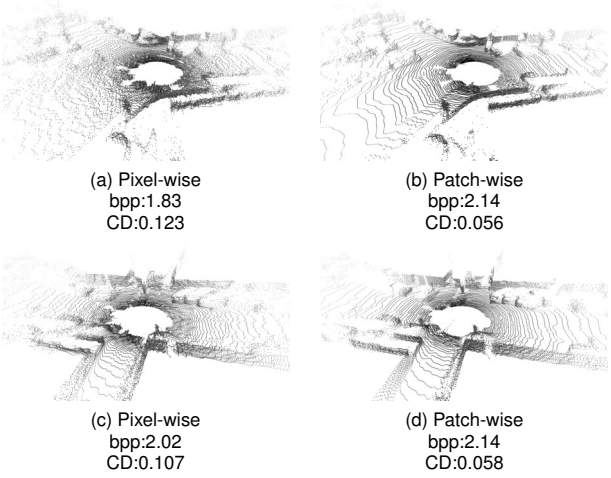
(b) Convergence speed.

FIGURE 6: Patch-wise vs. pixel-wise.

shown in Table 2, INR-based compression drastically reduces decoding latency, which is advantageous for on-demand, quality-driven services.

We note that the proposed scheme has a trade-off between reconstruction quality and encoding latency depending on the learning rate schedule. Here, we use initial and minimum learning rates of 1×10^{-4} and 1×10^{-12} for quality-sensitive users. When we use initial and minimum learning rates of 1×10^{-6} and 1×10^{-8} for encoding, the encoding latency is reduced to 30 minutes with a slight degradation in reconstruction quality. This means the proposed scheme can select quality-oriented or latency-oriented configurations based on application requirements.

C. ABLATION STUDY



(a) Pixel-wise bpp:1.83 CD:0.123
(b) Patch-wise bpp:2.14 CD:0.056
(c) Pixel-wise bpp:2.02 CD:0.107
(d) Patch-wise bpp:2.14 CD:0.058

FIGURE 7: Snapshot of the reconstructed LiDAR point clouds in pixel-wise and patch-wise proposed methods. Here, (a)-(b) and (c)-(d) show the reconstructed point clouds of frames 00 and 25 of sequence 00, respectively.

1) IMPACT OF PATCH-WISE INR ARCHITECTURE

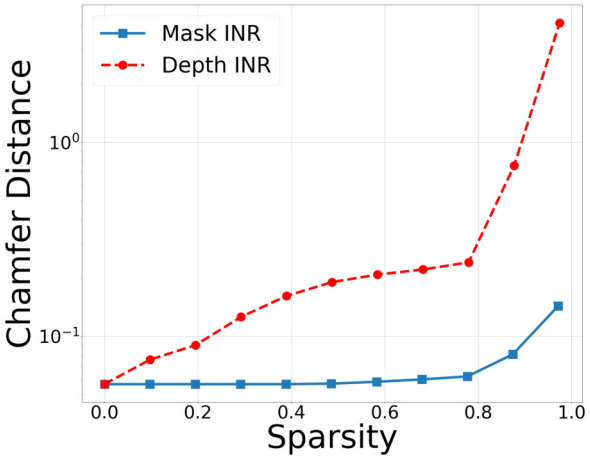
The proposed depth INR exploits the patch-wise architecture, whereas the pixel-wise architecture can be used for the depth INR. Fig. 6 (a) shows the CD of the proposed patch-wise INR and pixel-wise INR architectures as a function of bitrates under the different sequences of KITTI's LiDAR point cloud. We can see that the patch-wise depth INR achieves better CD than the pixel-wise architecture at large bitrate regimes in every LiDAR sequence. Specifically, BD-CD between the patch-wise and pixel-wise architectures is 0.047, 0.031, and 0.028 in frames 00, 25, and 50 of sequence 00, respectively. The effects on the visual quality are shown in Figs. 7 (a)–(d), respectively.

Fig. 6 (b) shows the CD performance of INR-based image compression methods as a function of the learning epochs. Our patch-wise architecture boosts the convergence speed by up to 13× compared to the pixel-wise architecture, and the fast convergence results in a short encoding delay.

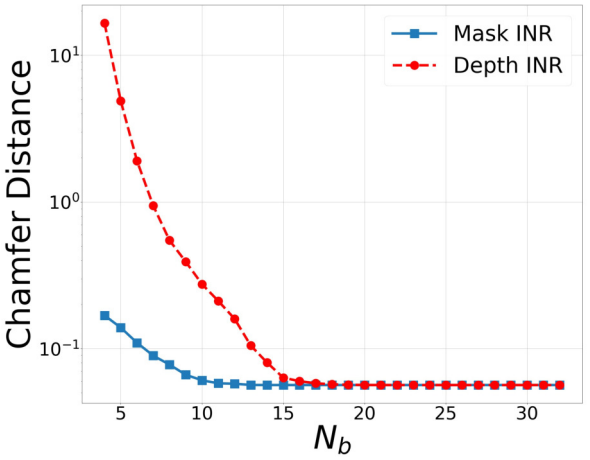
2) IMPACT OF MODEL COMPRESSION

After the encoder trains the depth and mask INR architectures, the trained weights are pruned and quantized for compression. Here, the proposed method can set different pruning ratios and bit depths for the depth and mask INRs. This section discusses the impact of model pruning and quantization on both INR model compression.

Figs. 8 (a) and (b) show the effect of model pruning and quantization for the depth and mask architectures. In pruning, the mask INR is similar in performance to the full model, although the sparsity is approximately 70%. However, pruning the model for the depth INR causes quality degradation even though the sparsity is only 10%. For quantization, a 16-bit model still retains almost the same CD as the original 32-bit model in depth INR, while the mask INR can reduce the



(a) CD for different pruning sparsity.



(b) Quantization with different N_b.

FIGURE 8: Model compression performance.

number of bits to 11.

3) IMPACT OF NETWORK ARCHITECTURE

This section discusses the effect of the configurations for the depth INR architecture, specifically the patch size N_p and layer size L, on the quality of the reconstructed LiDAR point cloud. The proposed depth INR uses the patch-wise architecture, and thus the depth image is divided into patches of size $N_p \times N_p$. Here, a small patch size increases the complexity of intra-patch learning, while a large patch size increases the complexity of inter-patch learning.

Fig. 9 shows CD performance varying N_p, and Fig. 10 demonstrates the corresponding snapshots of the reconstructed point cloud. We consider all the variants using the same model size in Fig. 8 with a sparsity of 0.0 and N_b of 32. The evaluation results demonstrated that the patch size of $N_p = 16$ yields the best CD performance. However, using

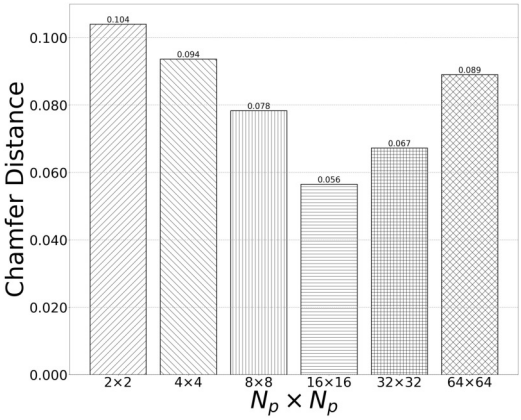
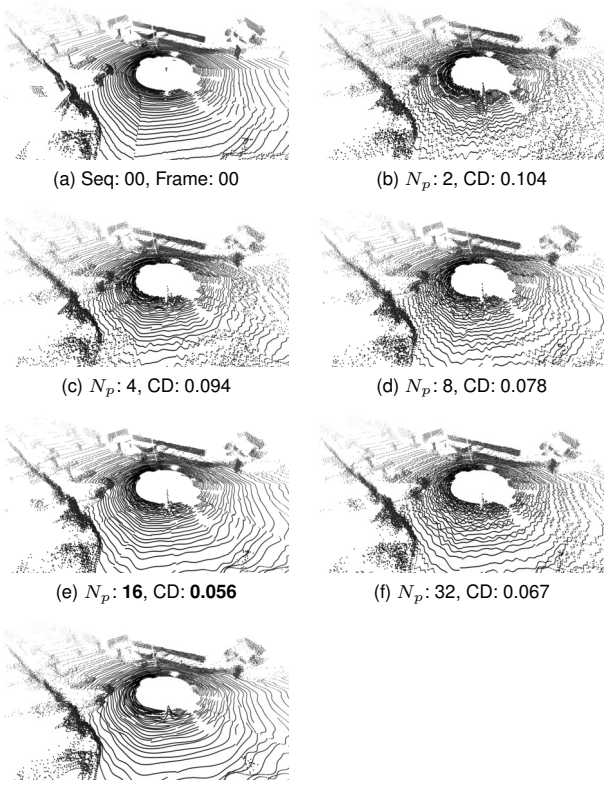


FIGURE 9: Chamfer distance under the different patch sizes.



(a) Seq: 00, Frame: 00
(b) N_p: 2, CD: 0.104
(c) N_p: 4, CD: 0.094
(d) N_p: 8, CD: 0.078
(e) N_p: **16**, CD: **0.056**
(f) N_p: 32, CD: 0.067
(g) N_p: 64, CD: 0.089

FIGURE 10: Snapshots of the reconstructed LiDAR point clouds in proposed methods under the different patch sizes $N_p \times N_p$. Here, (b)-(g) show the reconstructed point clouds of frame 00 of sequence 00.

larger or smaller N_p values leads to performance degradation. While a large N_p reduces the effectiveness of patch-wise modeling due to the limited pixel count, a small N_p requires covering a wider area. This makes it challenging to capture sharp depth transitions and preserve geometric details near boundaries.

Similarly, Fig. 11 and 12 show the 3D reconstruction quality of the proposed scheme and the corresponding snapshots

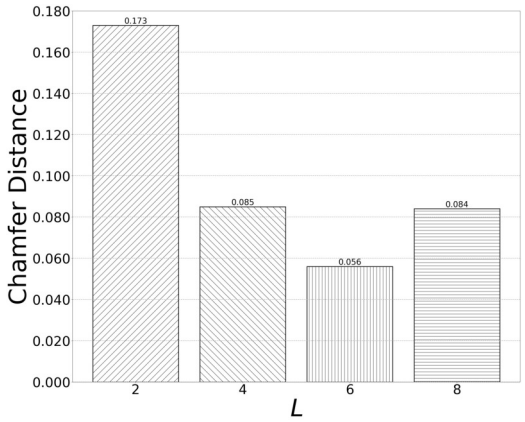


FIGURE 11: Chamfer distance under the different layer sizes.

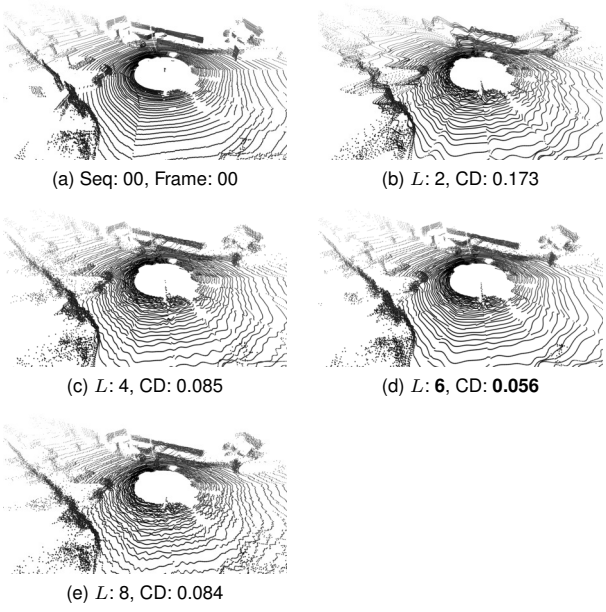


FIGURE 12: Snapshots of the reconstructed LiDAR point clouds in proposed methods under the different layer sizes L. Here, (b)-(e) show the reconstructed point clouds of frame 00 of sequence 00.

for different layer sizes L. The results indicate that a layer size of $L = 6$ is the most effective for CD performance.

V. CONCLUSION AND FUTURE WORK

We proposed a novel RI-based LiDAR point cloud compression method. The proposed method is designed to efficiently compress floating-point RIs using INR-based techniques and features a sophisticated architecture that combines separated learning for mask and depth images, patch-wise learning for depth images, and model compression. Experiments on the KITTI dataset show that the proposed method improves 3D reconstruction quality at low bitrates compared with conventional image codecs and representative baselines such as G-PCC, Draco, and COIN, and it also achieves strong 3D object detection accuracy in the low-bpp regime.

The proposed method has two limitations: encoding delay and transformation loss from RIs to 3D point clouds. While existing baselines require only a few milliseconds for encoding, implicit neural compression, including the proposed method, takes from tens of minutes to several hours. In summary, the proposed method's long encoding delay and short decoding delay make it well-suited for offline applications of LiDAR point clouds. To further reduce encoding latency, recent findings on learned initializations for coordinate-based neural representations [46] and meta-learned sparse INRs [47] can be integrated into our depth/mask INR architecture. We leave the implementation and evaluation of such integration as future work.

In addition, RIs with limited spatial resolution will lead to irreversible point loss during 2D-to-3D decoding, potentially degrading the performance of downstream tasks. In future work, we will consider integrating a point cloud generator [48] to obtain a denser point cloud from the limited resolution of RIs.

APPENDIX

This appendix provides further details for Table 1. Table 4 shows the detailed BD-CD performance across the different LiDAR frames.

ACKNOWLEDGMENT

This work was supported by JST·ASPIRE Grant Number JPMJAP2432.

REFERENCES

[1] K. Omasa, F. Hosoi, and A. Konishi, "3d lidar imaging for detecting and understanding plant responses and canopy structure," Journal of Experimental Botany, vol. 58, no. 4, pp. 881–898, 10 2006. [Online]. Available: https://doi.org/10.1093/jxb/erl142

[2] L. Jones and P. Hobbs, "The application of terrestrial lidar for geohazard mapping, monitoring and modelling in the british geological survey," Remote Sensing, vol. 13, no. 3, 2021. [Online]. Available: https://www.mdpi.com/2072-4292/13/3/395

[3] E. Kim and G. Medioni, "Urban scene understanding from aerial and ground lidar data," Machine Vision and Applications, vol. 22, no. 4, pp. 691–703, July 2011. [Online]. Available: https://doi.org/10.1007/s00138-010-0279-7

[4] X. Sun, S. Wang, M. Wang, Z. Wang, and M. Liu, "A novel coding architecture for LiDAR point cloud sequence," IEEE Robot. Autom. Lett., vol. 5, no. 4, pp. 5637–5644, 2020.

[5] G. K. Wallace, "The JPEG still picture compression standard," CACM, vol. 34, no. 4, p. 30–44, Apr. 1991.

[6] M. W. Marcellin, M. J. Gormish, A. Bilgin, and M. P. Boliek, "An overview of JPEG-2000," in DCC, 2000, pp. 523–541.

[7] E. Dupont, A. Goliński, M. Alizadeh, Y. W. Teh, and A. Doucet, "COIN: COmpression with implicit neural representations," in ICLR Workshop Neural Compression, 2021.

[8] V. Sitzmann, J. N. P. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," in NeurIPS, 2020, pp. 1–12.

[9] M. Tancik, P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron, and R. Ng, "Fourier features let networks learn high frequency functions in low dimensional domains," Adv. Neural Inf. Process. Syst., vol. 33, pp. 7537–7547, 2020.

TABLE 4: The list of BD-CD ↑ for the KITTI dataset across the different frames. Note that BD-CD is evaluated for each baseline using the proposed method as the reference. Positive values indicate that the proposed method achieves a lower chamfer distance than the corresponding baseline.

Seq.	Frame	JPEG†	JPEG2000†	HEIF†	AVIF†	COIN‡	G-PCC§	Draco§	Oct Attention §	R-PCC (Deflate) ¶	R-PCC (LZ4) ¶
00	00	1.197	0.661	0.465	0.260	1.039	0.099	0.146	-0.022	0.023	0.108
	25	1.277	0.719	0.490	0.237	0.969	0.071	0.146	-0.032	0.028	0.118
	50	1.588	0.923	0.607	0.266	1.264	0.081	0.153	-0.025	0.050	0.136
	75	1.696	1.036	0.661	0.270	0.983	0.068	0.153	-0.021	0.016	0.100
	100	1.208	0.667	0.450	0.247	1.261	0.088	0.189	-0.027	0.000	0.085
01	00	1.118	0.630	0.474	0.258	1.072	0.084	0.265	-0.036	-0.005	0.074
	25	1.040	0.581	0.462	0.285	1.073	0.091	0.292	-0.013	0.005	0.082
	50	1.128	0.624	0.485	0.271	1.113	0.121	0.238	-0.011	0.019	0.088
	75	1.411	0.754	0.546	0.244	0.969	0.115	0.177	-0.005	0.021	0.080
	100	1.472	0.842	0.606	0.239	1.250	0.075	0.199	-0.018	0.057	0.159
02	00	1.248	0.704	0.497	0.229	0.970	0.082	0.217	-0.056	-0.014	0.137
	25	1.037	0.561	0.419	0.250	0.986	0.063	0.215	-0.031	0.002	0.070
	50	1.039	0.558	0.418	0.266	0.996	0.099	0.272	-0.022	0.006	0.076
	75	1.101	0.607	0.458	0.273	0.968	0.075	0.203	-0.017	0.009	0.087
	100	1.176	0.629	0.446	0.250	1.276	0.095	0.223	-0.005	0.027	0.101
03	00	1.524	0.847	0.551	0.228	1.037	0.062	0.164	-0.031	0.036	0.124
	25	1.559	0.973	0.615	0.244	0.960	0.061	0.139	-0.040	0.019	0.109
	50	1.293	0.737	0.496	0.222	1.215	0.050	0.203	-0.055	-0.019	0.071
	75	1.129	0.560	0.405	0.232	0.927	0.060	0.202	-0.055	-0.041	0.036
	100	1.103	0.548	0.421	0.237	1.232	0.063	0.174	-0.045	-0.029	0.052
04	00	1.078	0.566	0.420	0.223	1.046	0.043	0.210	-0.072	-0.002	0.115
	25	1.348	0.806	0.546	0.239	1.045	0.085	0.199	-0.052	-0.012	0.065
	50	1.594	0.876	0.607	0.241	1.131	0.090	0.196	-0.020	0.028	0.105
	75	1.391	0.789	0.518	0.230	0.951	0.062	0.154	-0.044	0.005	0.089
	100	1.307	0.704	0.483	0.252	1.288	0.088	0.199	-0.005	0.030	0.109
05	00	1.108	0.620	0.439	0.252	1.007	0.083	0.179	-0.038	0.032	0.135
	25	1.198	0.661	0.464	0.256	1.009	0.077	0.187	-0.027	-0.005	0.071
	50	1.774	1.078	0.679	0.259	0.997	0.059	0.125	-0.043	0.002	0.084
	75	1.752	1.100	0.698	0.241	0.935	0.051	0.109	-0.058	-0.002	0.084
	100	1.589	0.924	0.601	0.233	1.258	0.055	0.166	-0.036	0.012	0.102
06	00	1.161	0.686	0.471	0.224	0.954	0.052	0.123	-0.069	0.010	0.122
	25	1.439	0.824	0.530	0.218	0.950	0.048	0.184	-0.068	-0.027	0.059
	50	1.359	0.814	0.531	0.226	0.951	0.046	0.159	-0.070	-0.031	0.045
	75	1.532	0.976	0.654	0.223	0.905	0.038	0.161	-0.079	-0.028	0.047
	100	1.128	0.623	0.451	0.230	1.205	0.082	0.141	-0.061	-0.045	0.035
Average		1.317	0.749	0.516	0.244	1.063	0.073	0.185	-0.037	0.005	0.090

†: Image based method(s), ‡: INR based method, §: Point Cloud based method(s), ¶: Range Image based method(s).

[10] A. Geiger, "Are we ready for autonomous driving? the kitti vision benchmark suite," in Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), ser. CVPR '12. USA: IEEE Computer Society, 2012, p. 3354–3361.

[11] 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, 2020.

[12] P. de Oliveira Rente, C. Brites, J. Ascenso, and F. Pereira, "Graph-based static 3D point clouds geometry coding," IEEE Trans. Multimed., vol. 21, no. 2, pp. 284–299, 2019.

[13] T. Fujihashi, T. Koike-Akino, T. Watanabe, and P. V. Orlik, "HoloCast+: hybrid digital-analog transmission for graceful point cloud delivery with graph fourier transform," IEEE Trans. Multimed., vol. 24, pp. 2179–2191, 2021.

[14] H. Kirihara, S. Ibuki, T. Fujihashi, T. Koike-Akino, and T. Watanabe, "Point cloud geometry compression using parameterized graph fourier transform," in Proceedings of SIGCOMM Workshop on Emerging Multimedia Systems, 2024, p. 52–57.

[15] S. Ueno, T. Fujihashi, T. Koike-Akino, and T. Watanabe, "Point cloud soft multicast for untethered XR users," IEEE Trans. Multimed., vol. 25, pp. 7185–7195, 2023.

[16] T. Fujihashi, S. Kato, and T. Koike-Akino, "Implicit neural representation for low-overhead graph-based holographic-type communications," in ICASSP, 2024, pp. 2825–2829.

[17] J. Kammerl, N. Blodow, R. B. Rusu, S. Gedikli, M. Beetz, and E. Steinbach, "Real-time compression of point cloud streams," in ICRA, 2012, pp. 778–785.

[18] X. Zhang and W. Gao, "Adaptive geometry partition for point cloud compression," IEEE Trans. Circuits Syst. Video Technol., vol. 31, no. 12, pp. 4561–4574, 2021.

[19] C. Fu, G. Li, R. Song, W. Gao, and S. Liu, "OctAttention: Octree-based large-scale contexts model for point cloud compression," AAAI, vol. 36, no. 1, pp. 625–633, Jun. 2022.

[20] L. Huang, S. Wang, K. Wong, J. Liu, and R. Urtasun, "Octsqueeze: Octree-structured entropy model for lidar compression," in CVPR, 2020, pp. 1313–1323.

[21] X. Zhou, C. R. Qi, Y. Zhou, and D. Anguelov, "RIDDLE: Lidar data compression with range image deep delta encoding," in CVPR, 2022, pp. 17 191–17 200.

[22] S. Wang, J. Jiao, P. Cai, and L. Wang, "R-PCC: A baseline for range image-based point cloud compression," in ICRA, 2022, pp. 10 055–10 061.

[23] C.-S. Liu, J.-F. Yeh, H. Hsu, H.-T. Su, M.-S. Lee, and W. H. Hsu, "BIRD-PCC: Bi-directional range image-based deep lidar point cloud compression," in ICASSP, 2023, pp. 1–5.

[24] L. Zhao, K.-K. Ma, Z. Liu, Q. Yin, and J. Chen, "Real-time scene-aware lidar point cloud compression using semantic prior representation," IEEE Trans. Circuits Syst. Video Technol., vol. 32, no. 8, pp. 5623–5637, 2022.

[25] H. Chen, B. He, H. Wang, Y. Ren, S.-N. Lim, and A. Shrivastava, "NeRV: Neural representations for videos," in NeurIPS, 2021.

[26] H. M. Kwan, G. Gao, F. Zhang, A. Gower, and D. Bull, "HiNeRV: Video compression with hierarchical encoding-based neural representation," in NeurIPS, 2023.

[27] J. C. Lee, D. Rho, J. H. Ko, and E. Park, "FFNeRV: Flow-guided

frame-wise neural representations for videos," in ACM-MM, 2023, p. 7859–7870.
[28] B. He, X. Yang, H. Wang, Z. Wu, H. Chen, S. Huang, Y. Ren, S.-N. Lim, and A. Shrivastava, "Towards scalable neural representation for diverse videos," in CVPR, 2023.
[29] Y. Xu, X. Feng, F. Qin, R. Ge, Y. Peng, and C. Wang, "VQ-NeRV: A vector quantized neural representation for videos," arXiv e-prints, Mar. 2024.
[30] S. R. Maiya, S. Girish, M. Ehrlich, H. Wang, K. Lee, P. Poirson, P. Wu, C. Wang, and A. Shrivastava, "Nirvana: Neural implicit representations of videos with adaptive networks and autoregressive patch-wise modeling," in CVPR, jun 2023, pp. 14 378–14 387.
[31] R. Xue, J. Li, T. Chen, D. Ding, X. Cao, and Z. Ma, "NeRI: Implicit neural representation of lidar point cloud using range image sequence," in ICASSP, 2024, pp. 8020–8024.
[32] H. Yan, Z. Ke, X. Zhou, T. Qiu, X. Shi, and D. Jiang, "DS-NeRV: Implicit neural video representation with decomposed static and dynamic codes," in CVPR, 2024, pp. 23 019–23 029.
[33] C. Gomes, R. Azevedo, and C. Schroers, "Video compression with entropy-constrained neural representations," in CVPR, 2023, pp. 18 497–18 506.
[34] H. Chen, M. Gwilliam, S.-N. Lim, and A. Shrivastava, "HNeRV: Neural representations for videos," in CVPR, 2023.
[35] X. Zhang, R. Yang, D. He, X. Ge, T. Xu, Y. Wang, H. Qin, and J. Zhang, "Boosting neural representations for videos with a conditional decoder," in CVPR, 2024.
[36] E. Dupont, H. Loya, M. Alizadeh, A. Goliński, Y. W. Teh, and A. Doucet, "COIN++: neural compression across modalities," TMLR, vol. 2022, no. 11, pp. 1–26, 2022.
[37] Y. Bai, C. Dong, C. Wang, and C. Yuan, "PS-NeRV: Patch-wise stylized neural representations for videos," in ICIP, 2023, pp. 41–45.
[38] O. D. Team, "Openpcdet: An open-source toolbox for 3d object detection from point clouds," https://github.com/open-mmlab/OpenPCDet, 2020.
[39] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "PointPillars: Fast encoders for object detection from point clouds," in CVPR, 2019, pp. 12 697–12 705.
[40] Y. Yan, Y. Mao, and B. Li, "Second: Sparsely embedded convolutional detection," Sensors, vol. 18, no. 10, p. 3337, 2018.
[41] S. Shi, X. Wang, and H. Li, "Pointrcnn: 3d object proposal generation and detection from point cloud," in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 770–779.
[42] A. Bazzi, M. Ying, O. Kanhere, T. S. Rappaport, and M. Chafii, "Isac imaging by channel state information using ray tracing for next generation 6g," IEEE Journal on Selected Topics in Electromagnetics, Antennas and Propagation, 2025, to appear.
[43] G. Bjøntegaard, "Calculation of Average PSNR Differences between RD-curves," TU-T SG16/Q6 Input Document VCEG-M33, 2001.
[44] "Draco 3D data compression," https://google.github.io/draco/, 2022.
[45] O. Devillers and P.-M. Gandoin, "Geometric compression for interactive transmission," in Proc. Inf. Vis. Conf., 2000, pp. 319–326.
[46] M. Tancik, B. Mildenhall, T. Wang, D. Schmidt, P. P. Srinivasan, J. T. Barron, and R. Ng, "Learned initializations for optimizing coordinate-based neural representations," in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2021, pp. 2846–2855.
[47] J. Lee, J. Tack, N. Lee, and J. Shin, "Meta-learning sparse implicit neural representations," in Advances in Neural Information Processing Systems, 2021.
[48] A. Kumar, C. Chen, A. Mian, N. Lobo, and M. Shah, "Sparse points to dense clouds: Enhancing 3D detection with limited LiDAR data," arXiv e-prints, Apr. 2024.

AKIHIRO KUWABARA received the B.S. degree from Osaka University, Osaka, Japan, in 2024. He is currently an M.S. student in the Graduate School of Information Science and Technology, Osaka University. His research interests include point cloud compression and delivery.

SORACHI KATO received B.E. and M.E. degrees from Osaka University, Japan, in 2021 and 2023, respectively. He is currently pursuing his Ph.D. degree in the Graduate School of Information Science and Technology, Osaka University, from April 2023. He is a research fellow (DC1) of Japan Society for the Promotion of Science from 2023. From 2023 to 2024, he was an intern at Mitsubishi Electric Research Labs. (MERL) working with the signal processing group. He received the Outstanding Paper Award from the Information Processing Society of Japan (JSPS) in 2022. His research interests are in the areas of RF sensing, deep neural signal processing, and multimedia neural compression.

PLACE PHOTO HERE

TOSHIAKI KOIKE-AKINO (M'05–SM'11) received the B.S. degree in electrical and electronics engineering, M.S. and Ph.D. degrees in communications and computer engineering from Kyoto University, Kyoto, Japan, in 2002, 2003, and 2005, respectively. During 2006–2010 he was a Postdoctoral Researcher at Harvard University, and is currently a Distinguished Research Scientist at Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA, USA. He received the YRP Encouragement Award 2005, the 21st TELECOM System Technology Award, the 2008 Ericsson Young Scientist Award, the IEEE GLOBECOM'08 Best Paper Award in Wireless Communications Symposium, the 24th TELECOM System Technology Encouragement Award, and the IEEE GLOBECOM'09 Best Paper Award in Wireless Communications Symposium. He is a Fellow of Optica.

TAKUYA FUJIHASHI received his B.E. degree in 2012 and his M.S. degree in 2013 from Shizuoka University, Japan. In 2016, he received his Ph.D. degree from the Graduate School of Information Science and Technology, Osaka University, Japan. He is currently an Assistant Professor at the Graduate School of Information Science and Technology, Osaka University since April 2019. He was a research fellow (PD) of Japan Society for the Promotion of Science in 2016. From 2014 to 2016, he was a research fellow (DC1) of Japan Society for the Promotion of Science. From 2014 to 2015, he was an intern at the Mitsubishi Electric Research Labs. (MERL) working with the Electronics and Communications group. He was selected as one of the Best Paper candidates in IEEE ICME (International Conference on Multimedia and Expo) 2012. His research interests are in the area of video compression and communications, with a focus on multi-view video coding and streaming.