IEEE Robotics and Automation Letters (RAL) paper presented at the
2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

A Novel Coding Architecture for LiDAR Point Cloud Sequence

Xuebin Sun[1]*, Sukai Wang[2]*, Miaohui Wang[3], Zheng Wang[4] and Ming Liu[2], *Senior Member, IEEE*

***Abstract*— In this paper, we propose a novel coding architecture for LiDAR point cloud sequences based on clustering and prediction neural networks. LiDAR point clouds are structured, which provides an opportunity to convert the 3D data to a 2D array, represented as range images. Thus, we cast the LiDAR point clouds compression as a range images coding problem. Inspired by the high efficiency video coding (HEVC) algorithm, we design a novel coding architecture for the point cloud sequence. The scans are divided into two categories: intra-frames and inter-frames. For intra-frames, a cluster-based intra-prediction technique is utilized to remove the spatial redundancy. For inter-frames, we design a prediction network model using convolutional LSTM cells, which is capable of predicting future inter-frames according to the encoded intra-frames. Thus, the temporal redundancy can be removed. Experiments on the KITTI data set show that the proposed method achieves an impressive compression ratio, with 4.10% at millimeter precision. Compared with octree, Google Draco and MPEG TMC13 methods, our scheme also yields better performance in compression ratio.**

I. Introduction

Advances in autonomous driving technology have widened the use of 3D data acquisition techniques. LiDAR is almost indispensable for outdoor mobile robots, and plays a fundamental role in many autonomous driving applications such as localization , path planning [1], and obstacle detection [2], etc. The enormous volume of LiDAR point cloud data could be an important bottleneck for transmission and storage. Therefore, it is highly desirable to develop an efficient coding algorithm to satisfy the requirement of autonomous driving.

Octree methods have been widely researched for point cloud compression. The main idea of octree-based coding methods is to recursively subdivide the current data according to the range of coordinates from top to bottom, and gradually form an octree adaptive structure. Octree method can hardly compress LiDAR data into very small volumes with low information loss. The vehicle-mounted LiDAR data is structured, which provides a chance to convert them into a 2D panorama range image. Some researchers focus on using image-based coding methods to compress the point cloud data. However, these methods are unsuitable for unmanned vehicles. Traditional image or video encoding algorithms, such as JPEG2000 , JPEG-LS [3], and HEVC [4], were designed mostly for encoding integer pixel values, and using them to encode floating-point LiDAR data will cause significant distortion. Furthermore, the range image is characterized by sharp edges and homogeneous regions with nearly constant values, which is quite different from textured video. Thus, coding the range image with traditional tools such as the block-based discrete cosine transform (DCT) followed by coarse quantization can result in significant coding errors at sharp edges, causing a safety hazard in autonomous driving.

In this research, we address the LiDAR point cloud sequence compression problem. Learning from the HEVC architecture, we propose a novel coding architecture for LiDAR point cloud sequences, which mainly consists of intra-prediction and inter-prediction technologies. For-intra-frames, we utilize a cluster-based intra-prediction method to remove the spatial redundancy. There are great structural similarities between adjacent point clouds. For inter-frames, we train a prediction neural network, which is capable of generating the future inter-frames using the encoded intra-frames. The intra- and inter-residual data is quantified and coded using lossless coding schemes. Experiments on the KITTI dataset demonstrate our method yields an impressive performance.

In our previous paper [5], an efficient compression algorithm for a single scan is developed based on clustering. Based on this previous technique, we propose a novel coding architecture for LiDAR point cloud sequences in this work. The contributions of the paper are summarized as follows.

- Learning from the HEVC algorithm, we develop a novel coding architecture for LiDAR point cloud sequences.
- For inter-frames, we design a prediction network model using convolutional LSTM cells. The network model is capable of predicting future inter-frames according to the encoded intra-frames.
- The coding scheme is specially designed for LiDAR point cloud sequences for autonomous driving. Compared with octree, Google Draco and MPEG TMC13 methods, our method yields better performance.

II. Related work

The compression of 3D point cloud data has been widely researched in literature. According to the types of point cloud data, compression algorithms can be roughly classified into four categories.

* The first two authors contributed equally to this work.
[1] Xuebin Sun is now with the Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hong Kong. He contributed to this work during his time at HKUST. (email: sunxuebin@tju.edu.cn; xuebinsun@cuhk.edu.hk)
[2] Sukai Wang and Ming Liu are with the Department of Electronic & Computer Engineering, Hong Kong University of Science and Technology, Hong Kong. (email: swangcy@connect.ust.hk; eelium@ust.hk)
[3] Miaohui Wang is with the College of Electrical and Information Engineering, Shenzhen University, China. (email: wang.miaohui@gmail.com)
[4] Zheng Wang is with the Department of Mechanical Engineering, the University of Hong Kong, Hong Kong, also with the Department of Mechanical and Energy Engineering, Southern University of Science and Technology, China. (email: wangz@sustech.edu.cn)

Structured single point cloud compression: Some researchers have focused on developing compression methods for structured LiDAR point cloud data. Houshiar *et al.* [6] propose an image-based 3D point cloud compression method. They map the 3D points onto three panorama images, and use an image coding method to compress the images. Similar to their approach, Ahn *et al.* [7] introduce an adaptive range image coding algorithm for the geometry compression of large-scale 3D point clouds. They explore a prediction method to predict the radial distance of each pixel using previously encoded neighbors, and only encode the resulting prediction residuals. In contrast, Zanuttigh *et al.* [8] focus on efficient compression of depth maps of RGB-D data. They develop a segmentation method to identify the edges and main objects of a depth map. After that, an efficient prediction process is performed according to the segmentation result, and the residual data between the predicted and real depth map is calculated. Finally, the few prediction residuals are encoded by conventional image compression methods.

Unstructured singe point cloud compression: Elseberg *et al.* [9] propose an efficient octree data structure to store and compress 3D data without loss of precision. Experimental results demonstrate their method is useful for an exchange file format, fast point cloud visualization, sped-up 3D scan matching, and shape detection algorithms. Golla *et al.* [10] present a real-time compression algorithm for point cloud data based on local 2D parameterizations of surface point cloud data. They use standard image coding techniques to compress the local details. Zhang *et al.* [11] introduce a clustering- and DCT-based color point cloud compression method. In their method, they use the mean-shift technique to cluster 3D color point clouds into many homogeneous blocks, and a clustering-based prediction method to remove spatial redundancy of point cloud data. Tang *et al.* [12] present an octree-based scattered point cloud compression algorithm. Their method improves the stop condition of segmentation to ensure appropriate voxel size. Additionally, using their method, the spatial redundancy and outliers can be removed by traversal queries and bit manipulation.

Structured point cloud sequence compression: Kammerl *et al.* [10] introduce a novel lossy compression method for point cloud streams to remove the spatial and temporal redundancy within the point data. Octree data structures are used to code the intra point cloud data. Additionally, they develop a technique for contrasting the octree data structures of consecutive point clouds. By encoding structural differences, spatial redundancy can be removed. Tu *et al.* [13] propose an image-based compression method for LiDAR point cloud sequences. They convert the LiDAR data losslessly into range images, and then use the standard image or video coding techniques to reduce the volume of the data. As image-based compression methods hardly utilize the 3D characteristics of point clouds, Tu *et al.* propose a SLAM-based prediction for continuous point cloud data in their follow-up work [14]. Experimental results show that the SLAM-based method outperforms image-based point cloud compression methods. In [15], Tu *et al.* develop a recurrent neural network with residual blocks for LiDAR point cloud streams compression. Their network structure is like a coding and decoding process. The original point cloud data is encoded into low-dimensional features, which is treated as encoded bit stream. The decoding process is to decode these low-dimensional features to the original point cloud data. In [16], Tu *et al.* present a real-time point cloud compression scheme for 3D LiDAR sensor U-Net. Firstly, some frames are choosen as key frames (I-frame), then they use the U-net to interpolate the remaining LiDAR frames (P-frames) between the key frames.

Unstructured point cloud sequence compression: Saranya *et al.* [17] propose a real-time compression strategy on various point cloud streams. They perform an octree-based spatial decomposition to remove the spatial redundancy. Additionally, by encoding the structural differences of adjacent point clouds, the temporal redundancy can be removed. Thanou *et al.* [18] present a graph-based compression for dynamic 3D point cloud sequences. In their method, the time-varying geometry of the point cloud sequence is represented by a set of graphs, where 3D points and color attributes are considered as signals. Their method is based on exploiting the temporal correlation between consecutive point clouds and removing the redundancy. Mekuria *et al.* [19] introduce a generic and real-time time-varying point cloud coding approach for 3D immersive video. They code intra-frames with an octree data structure. Besides this, they divide the octree voxel space into macroblocks and develop an inter-prediction method.

Generally, the aforementioned approaches can significantly reduce the size of point cloud data, and are capable for some specific applications. However, few of them are specially designed for LiDAR point clouds data compression, so using them to encode LiDAR point clouds is inefficient. However, we can learn from their coding techniques, such as prediction [14], clustering [11] and registration [19]. In this paper, we propose a novel coding scheme for LiDAR point cloud sequences. Our method can largely remove spatial and temporal redundancy.

III. Overview of Point Clouds Coding Scheme

In this paper, we propose a hybrid encoding/decoding architecture (intra-prediciton, inter-prediction, and residual data coding) for LiDAR point cloud sequences. Fig. 1 illustrates the encoding and decoding flowcharts of the proposed method. The order arrangement of the intra- and inter-frames is illustrated in Fig. 2. The number of I frames and P frames can be defined by parameter m and n, respectively. For instance, if $m = 5$ and $n = 5$, the input data, in the form of a LiDAR data stream, will be formatted as "IIIIIPPPPPIIIIIPPPPP...".

The intra-frames will be coded using the intra-prediction mode, which is a spatial prediction within the frame, to remove the spatial redundancy. According to the encoded intra-frames, the inter-prediction module, a prediction neural network, is capable of inferring the future inter-frames [20].

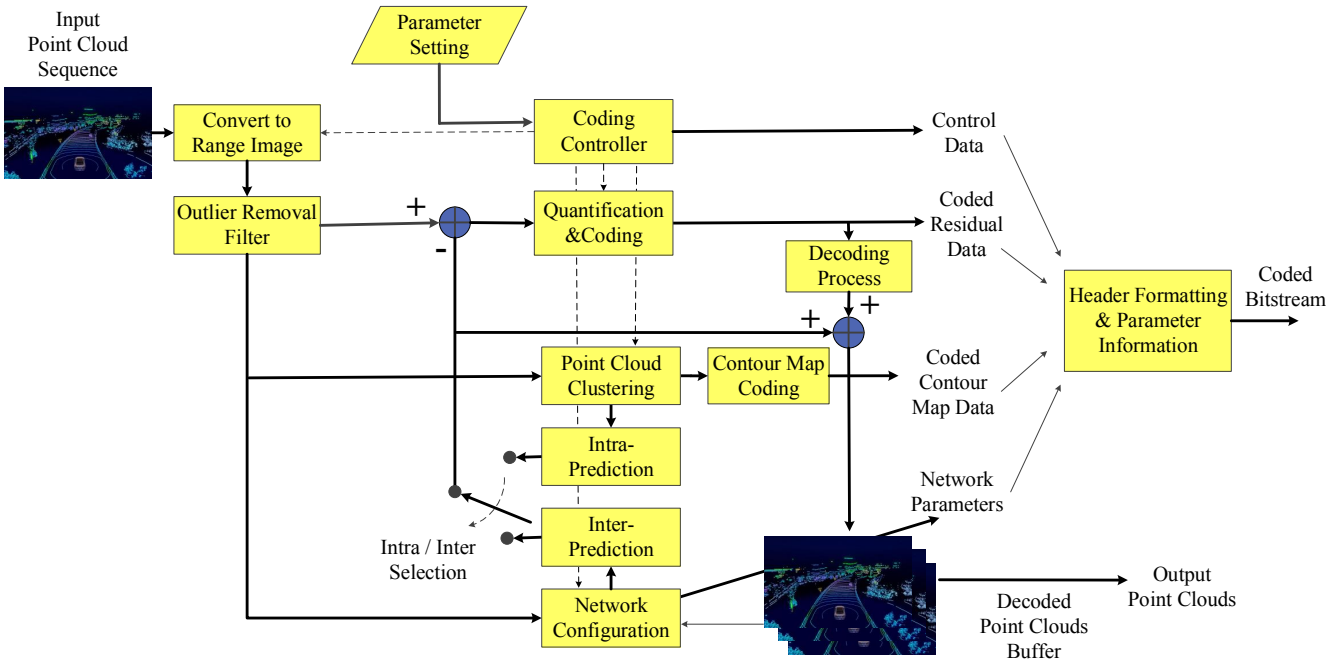

Fig. 1. Schematic of the LiDAR point cloud sequence compression codec

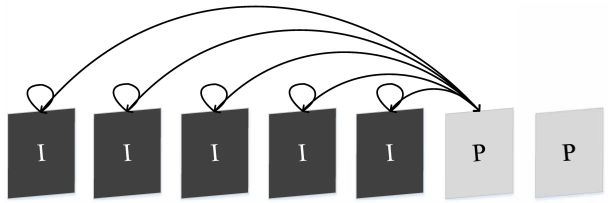

Fig. 2. Order arrangement of the intra- and inter-frames

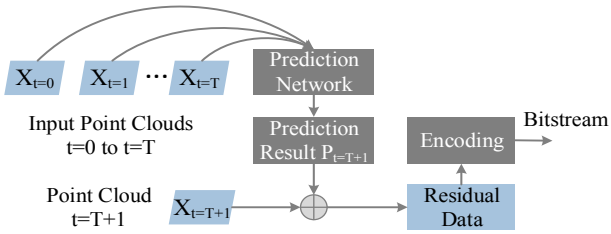

Fig. 3. Architecture of the inter-coding process

The residual signal of the intra- or inter-prediction, which is the difference between the original and prediction data, is encoded by lossless coding schemes. The coded control data, coded residual data, coded contour map data, and network parameters are packaged together in a certain way, forming the final bitstream.

Decoding is the inverse process of encoding. This is done by inverse scaling and decoding of the encoded data to produce the decoder approximation of the residual signal. This residual signal is then added to the prediction signal and forms the decoded point cloud. The final data representation, which is the duplicate of the possible output in the decoder, will be stored in a 'decoded point clouds buffer' and will be used for the prediction of subsequent point clouds. The components in the codec are briefly described as follows.

(a) Convert to Range Image: The point clouds are captured by Velodyne LiDAR HDL-64 sensors, which utilize 64 laser beams covering a vertical field of view of 26.9°and horizontal field of view of 360°. By coordinate system transformation, the 3D point cloud data can be converted into 2D grid arrays, known as panoramic range images.

(b) Outlier Removal Filter: To reduce the impact of outliers, a filter named Radius Outlier Removal is used [21]. Radius Outlier Removal calculates the number of adjacent points around each point and filters out the outliers.

(c) Point Cloud Clustering: For a single scan of a point cloud, points belonging to the same object have a lot of spatial redundancy. To eliminate the redundancy, a point cloud clustering technique is exploited to segment the range image into nonoverlapping clusters.

(d) Intra-Prediction: According to the clustering result, an efficient intra-prediction technique is performed to remove the spatial redundancy [5].

(e) Contour Map Coding: To recover the original point cloud data, we also need to encode the contour map. We divide the contour map into independent coding units and encode each unit with an integer value.

(f) Network Parameter Setting: The parameters of the neural network are configured according to the size of the input point cloud and the order arrangement of intra- and inter-frames.

(g) Inter-Prediction: A prediction neural network model is designed using convolutional LSTM cells. The model uses the encoded intra-frames to infer future inter-frames to remove the temporal redundancy.

(h) Residual Data Coding: The difference between the

Algorithm 1 The update process of the Prediction Network.

Require: The input t frames: $I = \{I_0, I_1, ..., X_t\}$;

Ensure:

The prediction k frames: $P = (P_{t+1}, P_{t+2}, ..., P_{t+k})$.

Assign the initial value: $E_l^t = 0$ ($l \in [0, L], t = 0$).
for $t = 0$ **to** T **do**
  for $l = L$ **to** 0 **do**
    if $t = 0, l = L$ **then**
      $O_L^t = ConvLSTM(E_L^{initial})$
    else if (t = 0) **then**
      $O_l^t = ConvLSTM(E_l^{initial}, UpSample(O_{l+1}^t))$
    else if (l = L) **then**
      $O_L^t = ConvLSTM(E_l^t, O_L^{t-1})$
    else
      $O_l^t = ConvLSTM(E_l^t, O_l^{t-1}, UpSample(O_{l+1}^t))$
    end if
  end for
  for $l = 0$ **to** $L - 1$ **do**
    if $l = 0$ **then**
      $P_0^t = Conv(O_0^t)$
      $E_l^t = P_l^t - x_t$
    else
      $P_l^t = ReLU(Conv(O_l^t))$
      $E_l^t = P_l^t - F_l^t$
    end if
    $F_{l+1}^t = MaxPool(Conv(E_l^t))$
  end for
end for

real and the predicted point cloud data is calculated as residual data. The residual data is quantified and encoded with lossless coding schemes.

(i) General Coder Control: The encoder uses pre-specified codec settings, including the precision configuration for module (b), cluster parameter configuration for module (c), network parameters configuration for module (g), and quantization parameter encoding method for module (h). In addition, it also controls the intra- and inter- frames, order arrangement.

(j) Header Formatting & Parameter Information: The parameter information, coded residual data, and coded contour map data are organized in a predefined order and form the final bitstream.

IV. Inter-Prediction Using Convolutional LSTM

In this paper, we develop a prediction neural network using convolutional LSTM cells, which is capable of generating future point clouds according to the encoded frames. Fig. 3 gives the architecture of the inter-coding process. The prediction network obtains the encoded points clouds $X = \{X_{t=0}, X_{t=1}, ..., X_{t=T}\}$, and generates the next frame point cloud $P_{t=T+1}$. The difference between the real point cloud $X_{t=T+1}$ and the predicted result $P_{t=T+1}$ will be calculated, quantified and encoded as the inter-bitstream.

A. LSTM-based Inter-prediction Network

Deep learning algorithms have been widely used to solve supervised learning tasks. However, point cloud prediction, as unsupervised learning, remains a difficult challenge. Figure 4 illustrates the overall architecture of the proposed prediction network using convolutional LSTM. The network consists of a series of repeated convolutional LSTM modules that attempt to locally predict the input and then subtract the input from the actual input and pass it to the next layer. $X = \{X_0, X_1, ..., X_T\}$ represents the input range images from $t = 0$ to T, while the $P = \{P_0, P_1, ..., P_T, P_{T+1}\}$ denotes the predicted results. The network mainly consists of three types of model: the error representation (E_l^t), the convolutional LSTM layer ($ConvLSTM_l^t$), and the feature extraction layer (F_l^t). E_l^t represents the difference between P_l^t and F_l^t; F_l^t revevies the E_l^t and extracts high features; and $ConvLSTM_l^t$ receives the E_l^t, O_l^{t-1}, and O_{l+1}^t, and makes a prediction.

When the network is constructed, point cloud prediction is performed. Consider a series of range images converted from point clouds. The lowest ConvLSTM level gets the actual sequence itself, while the higher layers receive the representative. The error is computed by a convolution from the layer below. The update process is described in Algorithm 1. The status update is performed through two processes: a top-down process in which the O_l^t and P_l^t state are calculated as described in the following formula; and then a forward process is performed to calculate the error E_l^t and higher-level targets F_l^t. The last noteworthy detail is that E_l^0 is initialized to zero, which means that the initial prediction is spatially consistent due to the convolutional nature of the network.

$$O_l^t = \begin{cases} ConvLSTM(E_L^{initial}) & t = 0, l = L \\ ConvLSTM(E_l^{initial}, Up(O_{l+1}^t)) & t = 0 \\ ConvLSTM(E_l^t, O_l^{t-1}) & l = L \\ ConvLSTM(E_l^t, O_l^{t-1}, Up(O_{l+1}^t)) & others \end{cases} \tag{1}$$

$$P_l^t = \begin{cases} Conv(O_0^t) & t = 0 \\ ReLU(Conv(O_l^t)) & others \end{cases} \tag{2}$$

The model is trained to minimize the weighted sum of error cell activities. Explicitly, the training loss is formalized in the following equation, represented by the weighting factor λ_t for the time and λ_l for the layer. The loss per layer is equivalent to the $||E_l^t||$. The loss function is defined as follows:

$$Loss = \sum_{t=0}^{t=T} \sum_{l=0}^{l=L} \lambda_t \cdot \lambda_l \cdot ||E_l^t||. \tag{3}$$

B. Training and Evaluation

We train and evaluate the proposed prediction model using the KITTI dataset. The point clouds are converted to range images with the resolution of 64×2000 pixels. After this, the images are processed to be grayscale, with values normalized between 0 and 1. We use 5K point clouds for training, 1K

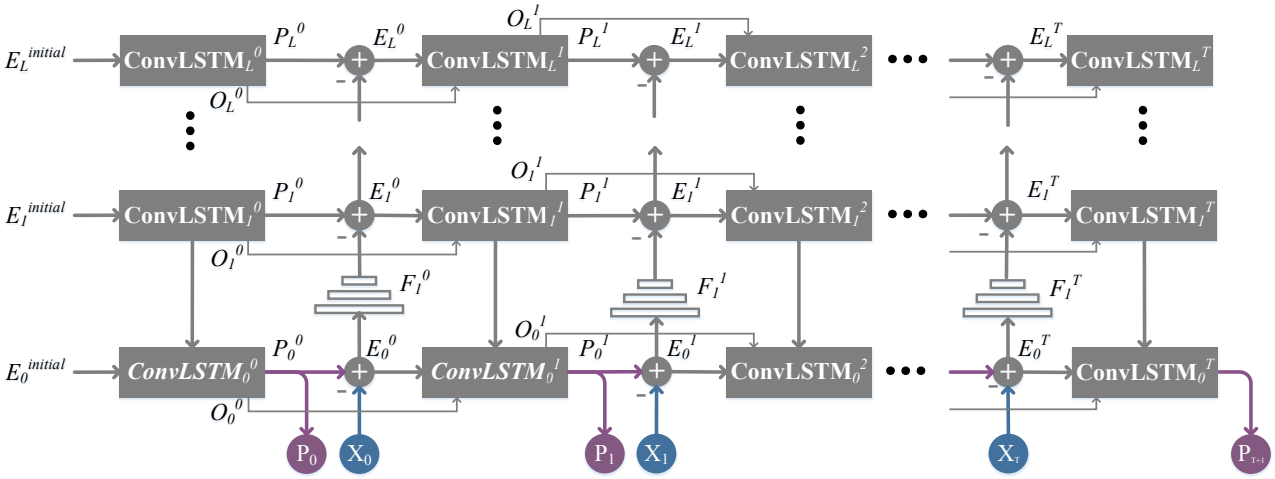

Fig. 4. Architecture of the point cloud prediction network.

for validation and 2K for testing. The point cloud data of the four scenes is equally distributed. The Adam method is used to train our model with a learning rate of 0.0001. We train our model with entropy optimization within 150 epochs. The batch size is set to 3, as limited by our GPU memory.

The model is capable of accumulating information over time to make accurate predictions of future frames. Since the representation neurons are initialized to zero, the prediction of the first time step is consistent. In the second time step, in the absence of motion information, the prediction is a fuzzy reconstruction of the first time step. After further iterations, the model adapts to the underlying dynamics to generate predictions that closely match the incoming frame.

V. Experimental Results

A. Evaluation Metrics

The main body of the proposed point clouds coding scheme is implemented in C++ using OpenCV [22], and PCL [23] libraries. The prediction network uses Keras 2.0.6 with CUDA 9.0 and cuDNN 7.0 libraries. The whole framework works on Intel 3.7GHz i7 CPU and a single GeForce GTX 1080Ti graphics card. We evaluated our framework on a series of experiments in KITTI dataset [24] in different scenes including campus, city, road and residuential, to demonstrate the generalization ability of our model.

The performance of the proposed method was evaluated in terms of compression rate (CR) and root mean square error (RMSE). The CR is the ratio between the compressed data size and the original one, defined in following formula [25]. The lower the value the better the performance.

$$Ratio = \frac{Compressed_{size}}{Original_{size}} \times 100\%, \tag{4}$$

The RMSE represents the square root of the corresponding points between the original point cloud and the reconstructed one. The original point cloud P_{input} is a set of K points, while $P_{decoded}$ represents the decoded point cloud with N points. K and N do not necessarily need to be equal.

$$\begin{aligned} P_{input} &= \{(p_i) : i = 0, ..., K-1\} \\ P_{decode} &= \{(p_i) : i = 0, ..., N-1\} \end{aligned}. \tag{5}$$

For each point in P_{input}, we take the distance to the nearest point $p_{nn-decode}$ in P_{decode}. The $p_{nn-decode}$ is efficiently computed via a K-d tree in the L2 distance norm. The RMSE is defined as follows:

$$\begin{aligned} MSR(P_{input}, P_{decode}) &= \frac{1}{K} \sum_{p_l \in P_{input}} ||p_l - p_{nn-decode}||_2^2 \\ MSR(P_{decode}, P_{input}) &= \frac{1}{N} \sum_{p_l \in P_{decode}} ||p_l - p_{nn-input}||_2^2 \\ RMSE &= \sqrt{\frac{MSR(P_{input}, P_{decode}) + MSR(P_{decode}, P_{input})}{2}} \end{aligned} \tag{6}$$

B. Coding Performance

Compression ratio for a single frame: Fig. 6 depicts the average compression ratios for a single frame with our proposed inter-prediction method combined with different lossless compression schemes, including Zstandard, LZ5, Lizard, Deflate, LZ4, BZip2, and Brotli. The point clouds are also directly encoded with these lossless coding schemes as the anchor. It should be noted that compared with coding the point cloud directly with the lossless schemes, the combination of the proposed inter-prediction method with lossless coding schemes achieves better performance. The best compression performance is achieved by the combination of the inter-prediction with the BZip2 method, with a compression ratio of 3.06% for the campus scene. The worst compression ratio is 6.92% for the residential scene coded by the combination of the proposed method with LZ4 scheme. Experimental results indicate that the inter-prediction method using the prediction network can effectively remove the time redundancy.

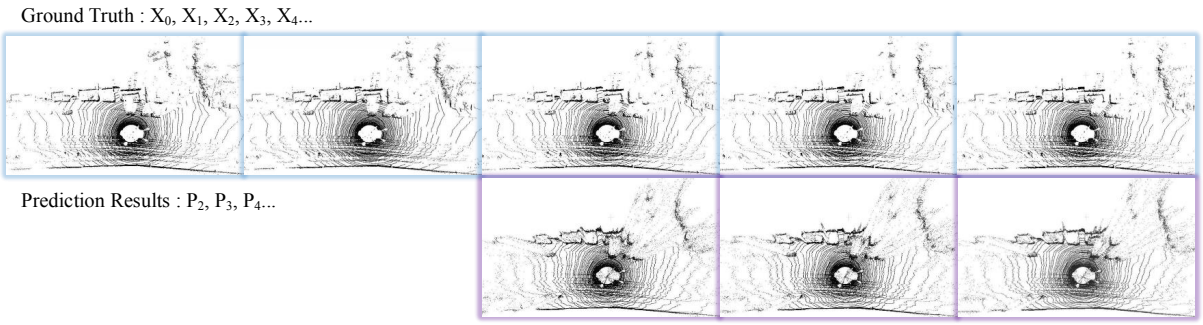

Fig. 5. Point clouds prediction results (residential scene).

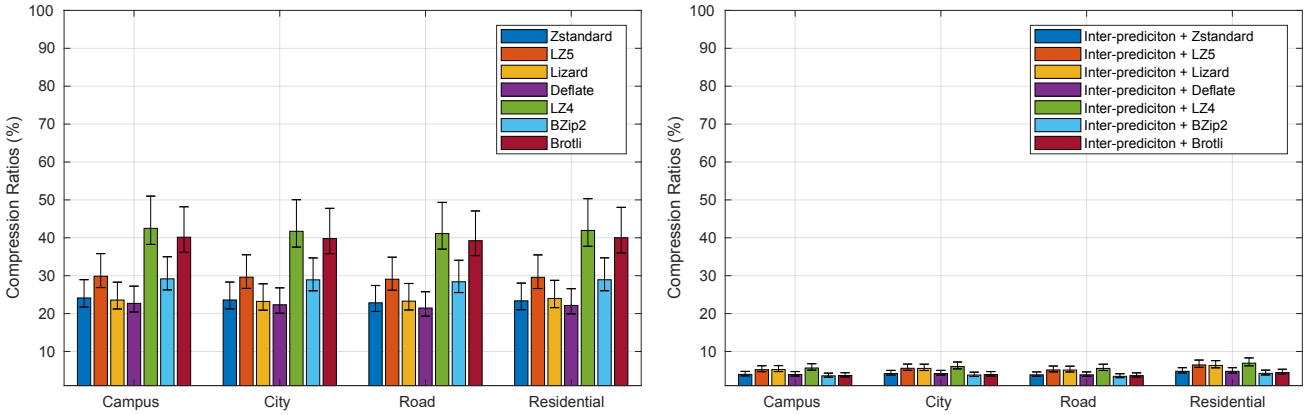

Fig. 6. Compression ratios of different lossless compression methods: (a) stand-alone lossless compression, (b) combination of the inter-prediction with lossless methods

Comparsion with recent methods: Table 1 describes the compression ratio results of the proposed method compared to the well-known octree method [28]. Considering that octree algorithms can not directly encode point cloud sequences, we assemble the point cloud sequence into a point cloud map by transforming each single point cloud into a global coordinate system.

$$\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{bmatrix} = R_{yaw} \times R_{pitch} \times R_{roll} \times \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} C_x \\ C_y \\ C_z \end{bmatrix}, \quad (7)$$

where $(\tilde{x}, \tilde{t}, \tilde{z})$ represents the coordinates after transformation; (x, y, z) denotes current coordinates; C_x, C_y, and C_z denote the translation matrix of the coordinates; and R_{yaw}, R_{pitch} and R_{roll} represents the rotation matrix of the yaw, pitch, and roll angle. These parameters can be obtained by the iterative closest point (ICP) algorithm [29].

The coding accuracy of the octree method is set to 1 cubic millimeter, 5 cubic millimeters, and 1 cubic centimeter, separately. In our experiments, we choose the BZip2 scheme to encode the residual data. The order of intra and inter-frames is formatted as IIIIIPPP...PPPIIIII.... Five intra-frames are encoded firstly, followed by fifteen interframes. From Table 2, it can be seen that our method outperform octree algorithm in compression ratio. However, octree method can process more than 1.28 million points per second. The complexity of our algorithm is higher than octree method. Additionally, octree has the advantage in searching operations, which is still indispensable for autonomous vehicle applications.

TABLE I

COMPRESSION RATE RESULTS COMPARED WITH THE OCTREE METHOD.

Scene	Proposed method Quantization Accuracy			Octree [28] Distance Resolution		
	$1mm$	$5mm$	$1cm$	$1mm^3$	$5mm^3$	$1cm^3$
Campus	3.94	3.14	2.52	21.27	8.05	5.75
City	4.06	3.22	2.64	23.98	10.76	8.40
Road	3.85	3.11	2.61	23.56	10.35	7.99
Residential	4.55	3.74	3.16	22.94	9.72	7.37
Average	**4.10**	**3.30**	**2.73**	**20.23**	**9.72**	**7.29**

To further verify the generality of the proposed prediction network, we also perform experiments using 32-lines LiDAR data captured by Velodyne HDL-32E sensors. Compression rate results are illustrated in Table 2. Experimental results demonstrate that the inter-coding method can also be used to encode 32-lines LiDAR data.

Figure 7 illustrates the performance of the proposed method compared with the cluster-based method [5], Draco [26], TMC13 [27], U-net-based method [16] and SLAM-based method [14]. As Tu *et al.* did not publish their coding performance for campus and city point clouds. For campus and city scenes, we only compare our method with cluster-based, Draco and TMC13 methods.

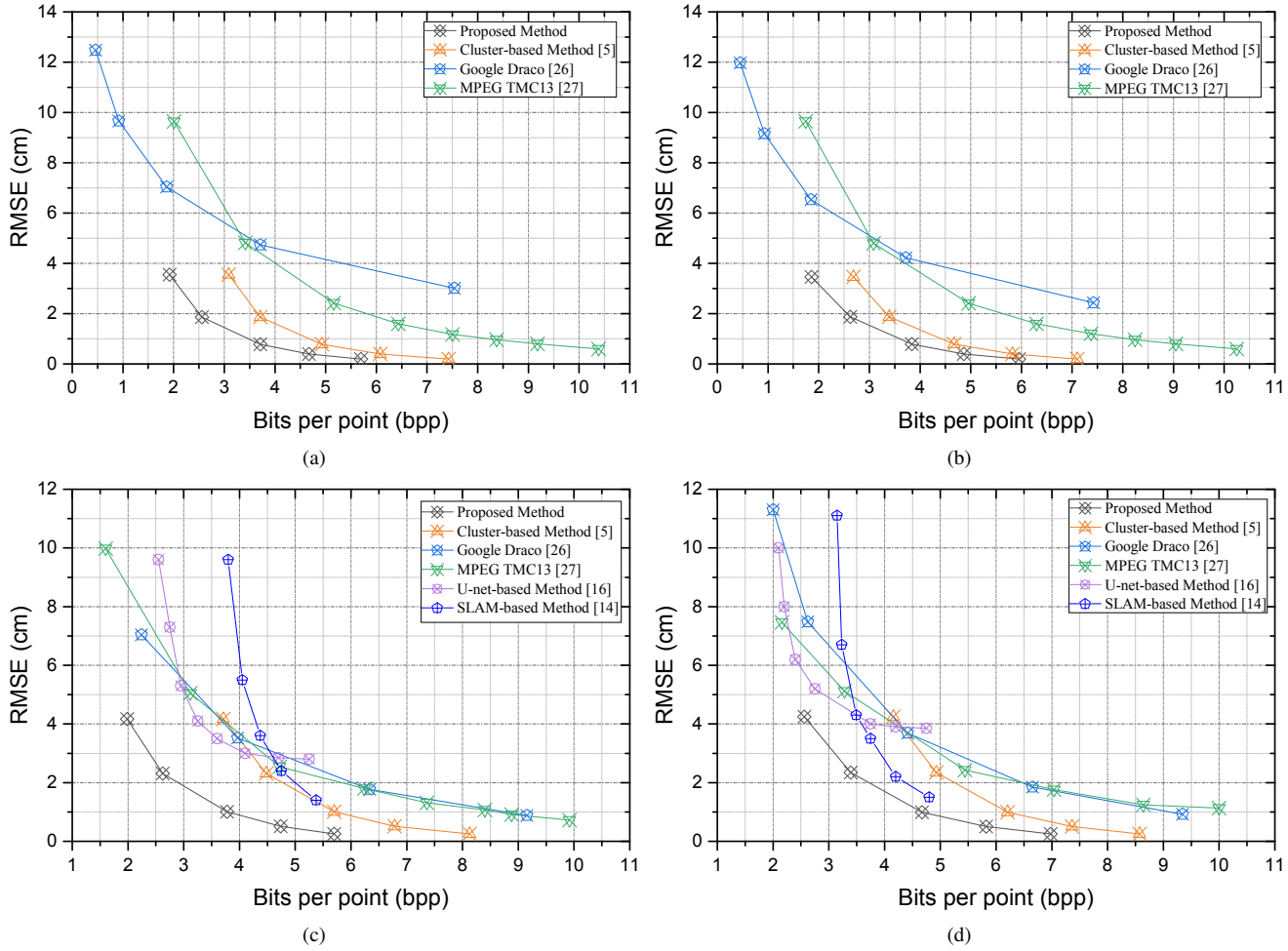

Fig. 7. RMSE-bpp curves performance of our method in comparision with cluster-based method [5], Google Draco [26], MPEG TMC13 [27], U-net-based method [16] and SLAM-based method [14]: (a) campus, (b) city, (c) road, (d) residential.

TABLE II

COMPRESSION RATE RESULTS ON 64-LINES AND 32-LINES LIDAR DATA.

Quantization Accuracy	1mm	5mm	1cm	2cm
64-lines	4.10	3.30	2.73	2.16
32-lines	4.26	3.53	2.93	2.41

TABLE III

AVERAGE ENCODING TIME FOR INTRA- AND INTER-FRAME (S).

Coding Time	Conversion	Prediciton	Encoding	Total
Intra-frame	0.110	0.071	0.023	0.204
Inter-frame	0.110	0.028	0.021	0.159

Before performing Draco algorithm, we have assemble the point clouds into a point cloud map. After that, we use the Google Draco method to compress the point cloud map and calculate the compression ratio per frame. Comparison results are given by RMSE-bpp curves, which reflect the relationship between the RMSE and bits per pixel (bpp). TMC13 is an emerging point cloud compression standard recently released at the 125th MPEG meeting. We experiment on four scenes of point clouds: campus, city, road and residential. A smaller RMSE and bpp mean better coding performance because they enable a lower RMSE with less bandwidth. It can be observed that our method obtains smaller bbp than Draco, TMC13 and cluster-based methods under similar RMSE reconstruction quality.

Speed performance: The intra-coding includes three steps, namely range image conversion, intra-prediction, and quantification & encoding. The inter-coding consists of range image conversion, inter-prediction, and quantification & encoding. We test the average speed performance of the proposed method with 500 frames. As illustrated in Tables 3, the total encoding time for intra- and inter-frame is 0.204s and 0.159s, respectively. Tables 4 depicts the average encoding and decoding time of the proposed method compared with octree, Draco, and TMC13 methods. It can be seen that the octree and Draco algorithm have low complexity. Our algorithm needs to be further optimized to meet the real-time requirement.

TABLE IV

ENCODING & DECODING TIME OF PROPOSED METHOD VERSUS OCTREE, DRACO, TMC13 (S).

Time	Intra-frame	Inter-frame	Octree	Draco	TMC13
Encoding	0.204	0.159	0.024	0.031	0.649
Decoding	0.013	0.034	0.009	0.010	0.337

VI. CONCLUSIONS AND DISCUSSION

In this paper, we proposed a novel and efficient compression architecture for the LiDAR point cloud sequence. Learning from the HEVC algorithm, we divided the frames in the point cloud sequence into intra-frames and inter-frames, which were encoded separately with different techniques. For intra-frames, we used the coding technique proposed in [5]. For inter-frames, we developed a prediction network using convolutional LSTM cells. The network can infer future inter-frames according to the encoded frames. Experimental results demonstrated that our approach significantly outperforms state-of-the-art techniques.

The drawback is that the proposed method can not apply to disordered point cloud compression, such as 3D human body sequences. Besides, the coding scheme can not satisfy real-time applications at present. Future studies will concentrate on improving its applicability and reducing the algorithm complexity with hardware acceleration.

REFERENCES

[1] M. Liu. Robotic Online Path Planning on Point Cloud. *IEEE Transactions on Cybernetics*, 46(5):1217–1228, May 2016.

[2] P. Yun, L. Tai, Y. Wang, C. Liu, and M. Liu. Focal Loss in 3D Object Detection. *IEEE Robotics and Automation Letters*, pages 1–1, 2019.

[3] Shaou-Gang Miaou, Fu-Sheng Ke, and Shu-Ching Chen. A lossless compression method for medical image sequences using JPEG-LS and interframe coding. *IEEE transactions on information technology in biomedicine*, 13(5):818–821, 2009.

[4] Miaohui Wang, King Ngi Ngan, Hongliang Li, and Huanqiang Zeng. Improved block level adaptive quantization for high efficiency video coding. pages 509–512, 2015.

[5] Xuebin Sun, Han Ma, Yuxiang Sun, and Ming Liu. A Novel Point Cloud Compression Algorithm Based On Clustering. *IEEE Robotics and Automation Letters*, pages 1–1, 2019.

[6] Hamidreza Houshiar and Andreas Nüchter. 3D point cloud compression using conventional image compression for efficient data transmission. In *2015 XXV International Conference on Information, Communication and Automation Technologies (ICAT)*, pages 1–8. IEEE, 2015.

[7] Jae-Kyun Ahn, Kyu-Yul Lee, Jae-Young Sim, and Chang-Su Kim. Large-scale 3D point cloud compression using adaptive radial distance prediction in hybrid coordinate domains. *IEEE Journal of Selected Topics in Signal Processing*, 9(3):422–434, 2015.

[8] Pietro Zanuttigh and Guido M Cortelazzo. Compression of depth information for 3D rendering. pages 1–4, 2009.

[9] Jan Elseberg, Dorit Borrmann, and Andreas Nüchter. One billion points in the cloud–an octree for efficient processing of 3D laser scans. *ISPRS Journal of Photogrammetry and Remote Sensing*, 76:76–88, 2013.

[10] Julius Kammerl, Nico Blodow, Radu Bogdan Rusu, Suat Gedikli, Michael Beetz, and Eckehard Steinbach. Real-time compression of point cloud streams. In *2012 IEEE International Conference on Robotics and Automation*, pages 778–785. IEEE, 2012.

[11] Ximin Zhang, Wanggen Wan, and Xuandong An. Clustering and dct based color point cloud compression. *Journal of Signal Processing Systems*, 86(1):41–49, 2017.

[12] Lin Tang, Fei-peng Da, and Yuan Huang. Compression algorithm of scattered point cloud based on octree coding. In *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, pages 85–89. IEEE, 2016.

[13] Chenxi Tu, Eijiro Takeuchi, Chiyomi Miyajima, and Kazuya Takeda. Compressing continuous point cloud data using image compression methods. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1712–1719. IEEE, 2016.

[14] Chenxi Tu, Eijiro Takeuchi, Chiyomi Miyajima, and Kazuya Takeda. Continuous point cloud data compression using SLAM based prediction. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1744–1751. IEEE, 2017.

[15] Chenxi Tu, Eijiro Takeuchi, Alexander Carballo, and Kazuya Takeda. Point cloud compression for 3d lidar sensor using recurrent neural network with residual blocks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3274–3280. IEEE, 2019.

[16] Chenxi Tu, Eijiro Takeuchi, Alexander Carballo, and Kazuya Takeda. Real-time streaming point cloud compression for 3d lidar sensor using u-net. *IEEE Access*, 7:113616–113625, 2019.

[17] S.thiruniral senthil R.Saranya. Real-time compression strategy on various point cloud streams. *International Journal of Computer Science and Mobile Computing*, 3(3):351–358, 2014.

[18] Dorina Thanou, Philip A Chou, and Pascal Frossard. Graph-based compression of dynamic 3D point cloud sequences. *IEEE Transactions on Image Processing*, 25(4):1765–1778, 2016.

[19] Rufael Mekuria, Kees Blom, and Pablo Cesar. Design, implementation, and evaluation of a point cloud codec for tele-immersive video. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(4):828–842, 2017.

[20] William Lotter, Gabriel Kreiman, and David Cox. Deep predictive coding networks for video prediction and unsupervised learning. *arXiv preprint arXiv:1605.08104*, 2016.

[21] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *2011 IEEE international conference on robotics and automation*, pages 1–4. IEEE, 2011.

[22] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008.

[23] Aitor Aldoma, Zoltan-Csaba Marton, Federico Tombari, Walter Wohlkinger, Christian Potthast, Bernhard Zeisl, Radu Bogdan Rusu, Suat Gedikli, and Markus Vincze. Tutorial: Point cloud library: Three-dimensional object recognition and 6 dof pose estimation. *IEEE Robotics & Automation Magazine*, 19(3):80–91, 2012.

[24] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[25] Lujia Wang, Luyu Wang, Yinting Luo, and Ming Liu. Point-cloud compression using data independent methoda 3d discrete cosine transform approach. In *2017 IEEE International Conference on Information and Automation (ICIA)*, pages 1–6. IEEE, 2017.

[26] Google. Draco: 3D Data Compression. *https://github.com/google/draco*, 2018.

[27] PCC WD G-PCC (Geometry-Based PCC). document iso/iec jtc1/sc29/wg11 mpeg. *N 17771, 3D Graphics*, Jul. 2018.

[28] Donald Meagher. Geometric modeling using octree encoding. *Computer graphics and image processing*, 19(2):129–147, 1982.

[29] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. International Society for Optics and Photonics, 1992.