A Novel Coding Architecture for Multi-Line LiDAR Point Clouds Based on Clustering and Convolutional LSTM Network

Xuebin Sun, Sukai Wang, *Graduate Student Member, IEEE*, and Ming Liu, *Senior Member, IEEE*

***Abstract*— Light detection and ranging (LiDAR) plays an indispensable role in autonomous driving technologies, such as localization, map building, navigation and object avoidance. However, due to the vast amount of data, transmission and storage could become an important bottleneck. In this article, we propose a novel compression architecture for multi-line LiDAR point cloud sequences based on clustering and convolutional long short-term memory (LSTM) networks. LiDAR point clouds are structured, which provides an opportunity to convert the 3D data to 2D array, represented as range images. Thus, we cast the 3D point clouds compression as a range image sequence compression problem. Inspired by the high efficiency video coding (HEVC) algorithm, we design a novel compression framework for LiDAR data that includes two main techniques: intra-prediction and inter-prediction. For intra-frames, inspired by the depth modeling modes (DMM) adopted in 3D-HEVC, we develop a clustering-based intra-prediction technique, which can utilize the spatial structure characteristics of point clouds to remove the spatial redundancy. For inter-frames, we design a prediction network model using convolutional LSTM cells. The network model is capable of predicting future inter-frames using the encoded intra-frames. As a result, temporal redundancy can be removed. Experiments on the KITTI dataset demonstrate that the proposed method achieves an impressive compression ratio (CR), with 4.10% at millimeter precision, which means the point clouds can compress to nearly 1/25 of their original size. Additionally, compared with the well-known octree, Google Draco, and MPEG TMC13 methods, our algorithm yields better performance in compression ratio.**

***Index Terms*— LiDAR, point cloud compression, clustering, convolutional LSTM.**

I. Introduction

ADVANCES in 3D data acquisition techniques have unleashed a new wave of innovation in many emerging applications, such as virtual/augmented reality (VR/AR), preservation of historical relics, 3D sensing for smart city, as well as autonomous driving. Especially for autonomous driving systems, LiDAR sensors play an indispensable role in a large number of key techniques, such as simultaneous localization and mapping (SLAM) [1], path planning [2], obstacle avoidance [3], and navigation. A point cloud consists of a set of individual 3D points, in accordance with one or more attributes (color, reflectance, surface normal, etc). For instance, the Velodyne HDL-64E LiDAR sensor generates a point cloud of up to 2.2 billion points per second, with a range of up to 120 m. This creates a great challenge for data transmission and storage. Thus, it is highly desirable to develop an efficient compression algorithm for LiDAR point cloud data.

Due to the characteristics of large scale, uneven distribution, and sparsity, compressing LiDAR data is a big challenge. The points in one frame merely have connectivity and topology, so it is hard to remove the spatial redundancies. Additionally, there is no explicit point correspondence between adjacent frames, which poses a challenge to remove the temporal redundancies. Octrees, as a data structure, have been widely researched in the last few decades to encode point clouds [4], [5]. The method is implemented by recursively dividing the 3D space into eight octants from top down. Octree-based coding technique merely takes into account the structural characteristics of LiDAR data captured by line-laser scanners. Using it to encode vehicle-mounted LiDAR data is inefficient.

By projecting 3D point clouds into 2D panoramic images, some researchers have focused on using image or video coding methods to compress LiDAR data [6]. Conventional image or video encoding algorithms are primarily used to encode integer pixel values. Using them to encode floating-point LiDAR data will cause distortion. Moreover, range images are characterized by sharp edges and uniform regions with almost equal values, which is quite different from textured videos. Using conventional tools such as block-based discrete cosine transform (DCT) and coarse quantization to encode distance images produces significant coding errors at sharp corners. These factors make it difficult to estimate the motion of points and predict the content of inter-frames using video coding strategies. However, since point cloud stream is continuous, we believe that it is possible to estimate the point motion, though new techniques need to be developed. The emerging deep learning method, known for

Manuscript received October 20, 2019; revised April 23, 2020 and September 18, 2020; accepted September 30, 2020. This work was supported in part by the National Natural Science Foundation of China under Grant U1713211, in part by the Collaborative Research Fund by Research Grants Council Hong Kong under Project C4063-18G, and in part by the HKUST-SJTU Joint Research Collaboration Fund awarded to Prof. Ming Liu under Project SJTU20EG03. The Associate Editor for this article was J. Li. *(Xuebin Sun and Sukai Wang contributed equally to this work.) (Corresponding author: Ming Liu.)*

Xuebin Sun was with The Hong Kong University of Science and Technology (HKUST), Hong Kong 999077. He is now with the Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hong Kong (e-mail: sunxuebin@tju.edu.cn; xuebinsun@cuhk.edu.hk).

Sukai Wang and Ming Liu are with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong 999077 (e-mail: swangcy@connect.ust.hk; eelium@ust.hk).

Digital Object Identifier 10.1109/TITS.2020.3034879

its ability to learn features and patterns from data, may be a solution.

In this article, we propose a novel compression framework for LiDAR point cloud sequences. As LiDAR data is structured, we project the 3D point clouds to 2D range images. Thus, we cast the 3D point clouds compression as range images coding problem. Inspired by the HEVC algorithm [7], we design a coding architecture for the range images, which mainly consists of intra-coding and inter-coding. For intra-coding, we develop a clustering-based intra-prediction technique to remove spatial redundancy, while for inter-coding, we train a prediction neural network to infer the 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 [8], [9]. The major contributions of the paper are summarized as follows.

- Inspired by the HEVC algorithm, we develop a novel coding architecture for LiDAR point cloud sequences;
- To remove spatial redundancy, we propose an efficient intra-prediction method for intra point clouds using clustering and quadric surface fitting techniques;
- To remove temporal redundancy, we design a prediction neural network using convolutional the LSTM cell. The network is capable of predicting future inter-frames using the encoded intra-frames;
- The algorithm is specially designed for line-laser scanner data, meeting the requirements for autonomous driving. Compared with octree [10], Google Draco [11] and MPEG TMC13 [12], our method yields better performance.

The rest of this article is structured as follows. In Section II, we discuss related works. In Section III, we give an overview of the point cloud coding framework. The intra-coding method and inter-coding method are presented in Section IV and Section V, respectively. Experimental results are shown in Section VI. Finally, the paper is concluded along with possible future research directions in Section VII.

II. Related Work

Over the past decade, scholarly works on point cloud compression have been extensive. Taking the characteristics of the point cloud as a major consideration, these methods can be roughly classified into two categories, structured and unstructured point cloud compression, and each category can be further divided into static single frame and dynamic point cloud compression.

A. Structured Static Point Cloud Compression

Liu *et al.* [13] propose a scan-line-based lossless compression algorithm for point clouds. In their method, they employ a distance-based predictor to predict the forthcoming point using the information of previous points, and use an arithmetic coding scheme to code the residual data. Houshiar and Nüchter [14] propose an image-based compression method for 3D point clouds, in which they map the point cloud onto panoramic images, and use image compression methods to compress the generated panoramic images. Ahn *et al.* [15] present a geometry compression algorithm for large-scale 3D point clouds to encode radial distances in a range image. In their method, they design twelve prediction modes for radial distances, and only encode the prediction residuals using a context-based entropy coder. Some other methods focus on encoding RGB-D data. Zanuttigh and Cortelazzo [16] introduce a novel strategy for the compression of depth maps. They develop a segmentation approach to extract edges and main objects, and predict the surface shape from the segmented regions. Finally, the few prediction residuals are efficiently encoded by standard image compression algorithms. Morell *et al.* [17] propose a geometric 3D point cloud compression approach based on the geometric structure of man-made scenarios. In their method, the points of each scene plane are represented as a Delaunay triangulation and a set of points/area information.

B. Structured Dynamic Point Cloud Streams Compression

Yang *et al.* [18] develop a fast transmission method for 3D point clouds, in which they remove the redundancy information by using filters and segmentation algorithms. Wang *et al.* [19] propose a 3-D image warping-based compression method for RGB-D data. They combine egomotion estimation and 3-D image warping techniques and use a lossless coding approach to encode the depth data. Their method has the advantage of high speed and high compression ratio, and is capable in real-time applications. Cohen *et al.* [20] develop a compression method for organized point clouds, in which they map 3D point cloud data to a 2D array and adaptively fit them with hierarchical patches. Their method obtains better performance in compression ratio compared with the octree-based codec.

C. Unstructured Static Point Cloud Compression

Oliveira *et al.* [21] propose a graph-based lossy coding algorithm for the geometry of static point clouds. They use an octree-based technique for a base layer and a graph-based transformation technique for the enhancement layer, where residual data is coded. Experimental results show their method achieve impressive performance. Wang *et al.* [22] use 3D DCT to compress point-cloud data, achieving a high compression ratio and flexible reconstruction behaviors compared with related methods. Fan *et al.* [23] propose a point cloud geometry encoder based on hierarchical point clustering. Firstly, a hierarchy of level of detail (LOD) is constructed using the adapted Generalized Lloyd Algorithm (GLA). After that, they progressively encode the LOD hierarchy using effective representation, prediction, and entropy coding.

D. Unstructured Dynamic Point Cloud Streams Compression

Nguyen *et al.* [24] propose a compression method for human body sequences, in which they develop graph wavelet filter banks to time-varying geometry and color signals living on a mesh representation of the human body. Thanou *et al.* [25] develop a graph-based motion estimation and compensation scheme for dynamic 3D point cloud

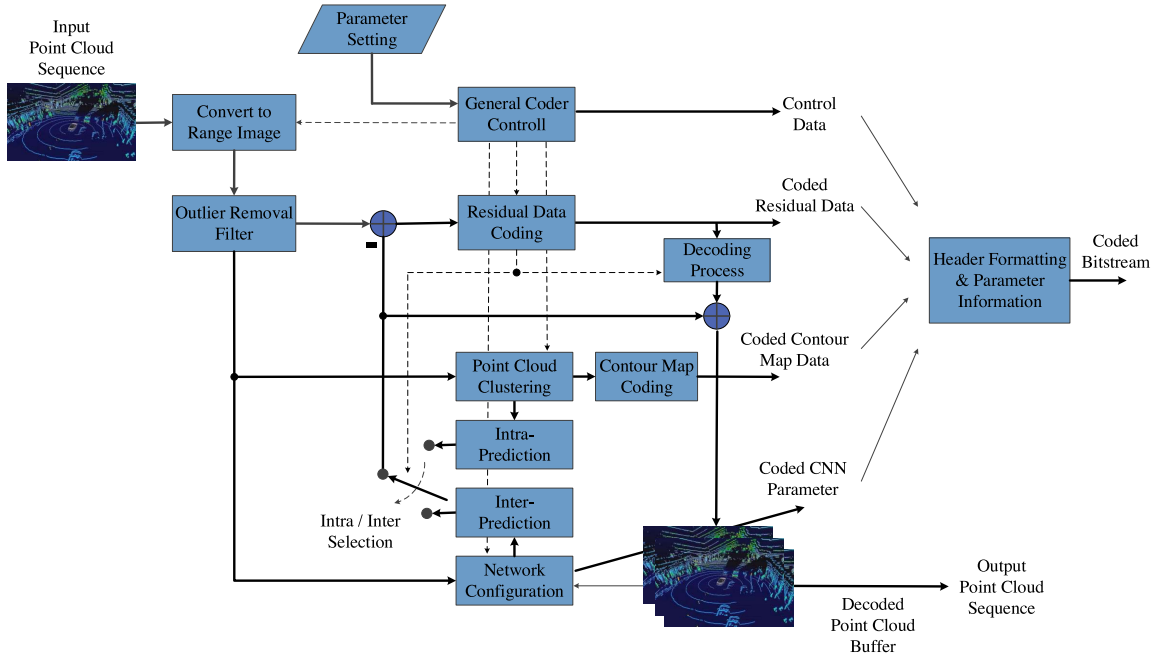


Fig. 1. Schematic of the LiDAR point cloud sequence compression codec.

sequences. In their method, the time-varying geometry of point cloud sequences is represented by a set of graphs, where 3D positions and color attributes are considered as signals. Then the motion is estimated on a sparse set of representative vertices based on new spectral graph wavelet descriptors. Queiroz *et al.* [26] introduce a novel motion-compensation approach to encoding dynamic voxelized point clouds (VPCs) at low bit rates. In their method, the VPC is divided into blocks and coded by intra-frames or replaced by a block in previous frame through motion compensation.

E. Summary and Analysis

Generally, the exiting algorithms can significantly reduce the point cloud data size and can be used for various applications, such as virtual reality, scanning of historical artifacts, and 3-D printing. However, few of them aim at compressing dynamically structured LiDAR point cloud streams. Octree methods are not ideal choice in autonomous driving because of the loss of points, while image-based compression methods fail to utilize the spatial geometric characteristics of point clouds, so their compression performance is inefficient. Fortunately, we can learn from their coding concepts, such as image-based coding technique [14], prediction concept [15] and clustering method [23]. In this article, we present a novel coding algorithm for LiDAR point cloud sequences based on clustering and a convolutional LSTM network.

III. Overview of Point Cloud Coding Framework

In this article, we propose a hybrid encoding/decoding architecture (intra-/inter- prediciton and residual data coding) for LiDAR point cloud sequences. Fig. 1 shows the coding and decoding flowcharts illustrating our proposed method [27]. The order arrangement of the intra- and inter frames- is

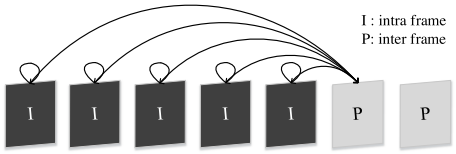


Fig. 2. The order arrangement of the intra- and inter-frames.

illustrated in Fig. 2. The number of I frame and P frame can be defined by parameter m and n. 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 I 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 I frames, the inter-prediction module, a neural network, is capable of inferring the future P frames [28]. The residual signal of the intra- or inter-prediction, which is the difference between the original and its prediction data, is encoded by lossless or lossy 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 coded 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 cloud buffer' and will be used for 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

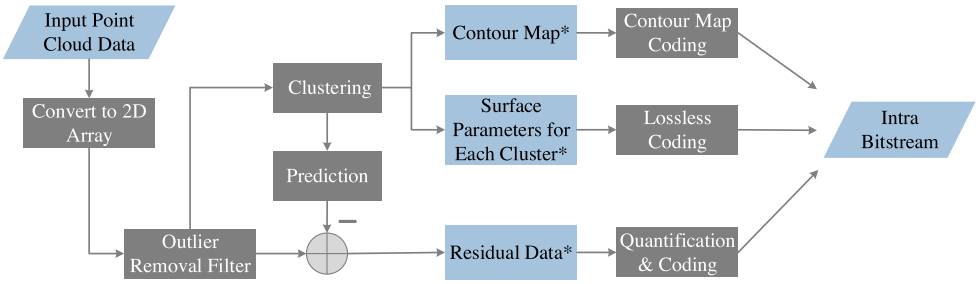


Fig. 3. Architecture of the intra-coding process.

utilize 64 laser beams covering a vertical field of view of 26.9°and horizontal field of view of 360°. The sensor represents the 3D geometry of a point by measuring the radial distance from the center of the scanner to an object in a uniformly distributed angular direction. By coordinate system transformation, the 3D point cloud data can be converted into 2D grid arrays, known as panoramic range image.

(b) Outlier Removal Filter: In fact, the LiDAR sensor produces a large number of abnormal values during range measurement. The outliers will reduce the efficiency of the algorithm with higher computing costs. To reduce the impact of outliers, a filter named Radius Outlier Removal is used [29]. 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. In order to eliminate the redundancy, a graph-based point cloud clustering technique is exploited to segment the range image into nonoverlapping clusters.

(d) Intra-Prediction: According to the clustering result, we develop an efficient intra-prediction technique based on quadric surface fitting. We perform quadric surface fitting for the clusters with a large number of points. According to the fitting surface and LiDAR parameters, the predicted value can be obtained.

(e) Contour Map Coding: In order to recover the original point cloud data, we also need to encode the contour map. In [30], Matejek *et al.* propose an efficient compression method for segmentation data in biomedical imaging. Inspired by their method, 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 real point cloud data and the predicted 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. Intra-Prediction Based on Clustering and Quadric Surface Fitting

Figure 3 gives the architecture of the intra-point-cloud coding process, where the grey blocks indicate processing steps and the blue blocks represent data. Particularly, the processed data, which need to be further encoded by lossless schemes, are marked with an asterisk. The workflow and example results are illustrated in Fig. 4. The intra-frame encoder consists of four main stages. Firstly, it converts the point cloud into a 2D range image and filters outliers. Secondly, a graph-based clustering technique is performed using the range image. Thirdly, based on the clustering result, we perform quadric surface fitting for each cluster to figure out the best fitting plane. According to the fitting surface and LiDAR parameters, a predicted range map can be computed. Thirdly, the difference between the predicted and the real range image is calculated as the residual data, which will be quantified and encoded by lossless schemes. To be able to rebuild the point cloud, the contour map and surface parameters also need to be saved. Finally, the encoded contour map data, surface parameters and residual data compose the intra-bitstream.

A. Pre-Processing

We experimented with the KITTI dataset captured by Velodyne HDL64 sensors, which represents the 3D geometry of the environment by rotating clockwise at a frequency of 10 Hz. While the sensor is rotating, all 64 lasers that are positioned vertically at a predetermined pitch angle emit simultaneously at different rotation angles $360°/n$, where n denotes the number of emission times during a 360° rotation. The angular

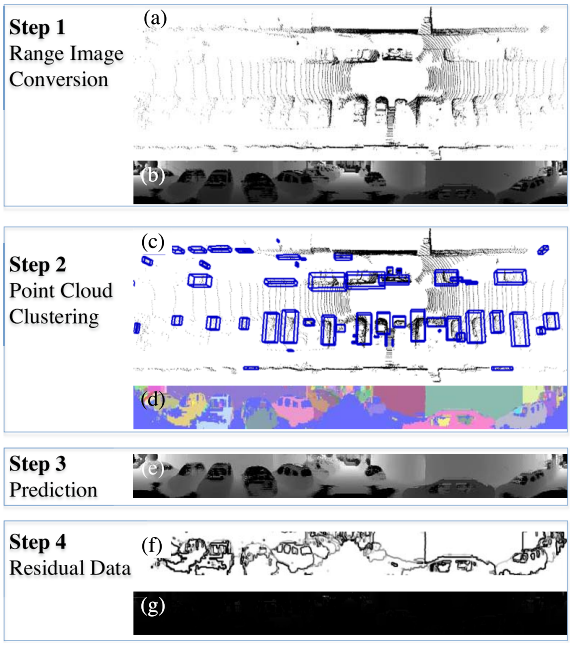


Fig. 4. The overall workflow of the intra-prediction method: (a) input point cloud; (b) range image; (c) clustering result in the point cloud; (d) segmentation result in the range image; (e) prediction map; (f) contour map; and (g) residual data.

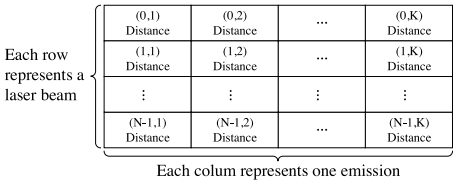


Fig. 5. Convert LiDAR point cloud to 2D array.

resolution of the Velodyne HDL-64 in the direction of rotation is approximately 0.18°, while in the vertical direction, the pitch angle difference of adjacent lasers is about 1.33°. This makes the point cloud denser in the lateral direction and relatively sparse in the radial direction. The scan data thus can be converted to a 2D array, as illustrated in Fig. 5. After that, the isolated points are filtered out [29]. The original point cloud and the transformed range image results are shown in Fig. 4 (a) and (b), respectively.

B. Graph-Based Clustering

We use graph-based techniques to complete the segmentation. According to the range image, the points are represented as a weighted undirected graph $G = (V, E)$, where V is the set of nodes representing points and E is the set of edges between adjacent points. Given the point cloud $P = (p_1, p_2, p_3, \ldots)$, we construct the problem of segmenting the set into disjoint subsets s_i s.t. $\cup_i s_i = V$.

We exploit a 4-connected graph for segmentation, which is extracted from the laser scans. Each node is connected to four adjacent others by edges labeled with three edge weights. This means for an edge $e_{i,j} =< \omega_\alpha, \omega_\beta, \omega_{(u,v)} >$ to connect two vertexes. ω_α represents the angle between two

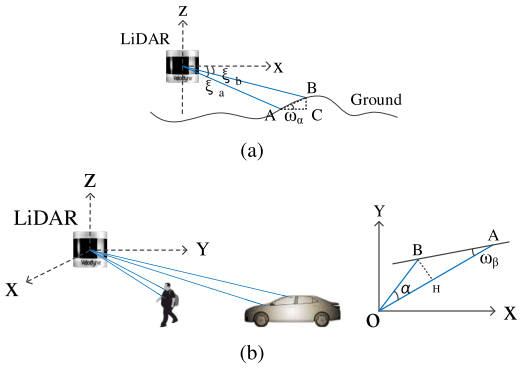


Fig. 6. Schematic diagram of the clustering technique: (a) ground extraction fundament; and (b) clustering fundament.

adjacent points in the vertical direction with the xoy plane, while ω_β indicates the angle between two neighboring beams, as depicted in Fig. 6. $\omega_{(u,v)}$ is the Euclidean distance in the three-dimensional space between to adjacent points. According to Fig. 6, ω_α, ω_β and $\omega_{(u,v)}$ are defined as follows:

$$\begin{aligned} \omega_\alpha &= arctan(|BC|, |AC|) = arctan(\Delta z, \Delta x) \\ \Delta z &= |R_{r-1,c} sin\zeta_\alpha - R_{r,c} sin\zeta_\beta| \\ \Delta x &= |R_{r-1,c} cos\zeta_\alpha - R_{r,c} cos\zeta_\beta|, \qquad (1) \\ \omega_\beta &= arctan\frac{|BH|}{|HA|} = arctan\frac{d_2 sin\alpha}{d_1 - d_2 cos\alpha}, \qquad (2) \\ \omega_{(u,v)} &= \sqrt{(u_x - v_x)^2 + (u_y - v_y)^2 + (u_z - v_z)^2}, \qquad (3) \end{aligned}$$

where ζ_α and ζ_β denote the vertical angles of the laser beams corresponding to row $r-1$ and r, respectively. d_1 and d_2 denote the distance between OA and OB, respectively, and α represents the angle between two beams. The ζ_α, ζ_β, and α can be obtained by LiDAR sensor parameters.

ω_α is used to extract the point cloud ground, while ω_β and $\omega_{(u,v)}$ are used for object clustering. As can be seen from Fig. 6 (a), if point A and point B belong to the ground, the angle ω_α is very small ($\omega_\alpha < thres_\alpha$), and vice versa. The threshold $thres_\alpha$ is predefined. Fig. 6 (b), if the two points A and B belong to the same object, then A and B are close, and the angle ω_β is close to 90°. Otherwise, if points A and B come from different objects, they are farther apart, and the angle ω_β is close to 0°or 180°. Additionally, the Euclidean distance is smaller for the points belonging to the same object. Two points merge into the same segment according to the conditions $|90° - \omega_\beta| < thres_\beta$, $\omega_{(u,v)} < thres_{(u,v)}$. In our experiment, $thres_\alpha$ and $thres_\beta$ are set to 15°and 10°, respectively. $thres_{(u,v)}$ is a changeable value along the radius direction of the LiDAR center. Due to the radiation of the LiDAR data, the distance threshold of the farther point cloud should be larger. We set the minimal distance threshold as 0.2 m and add 0.025m to it for each 2 meters far away.

Given the graph of the point cloud and the three edge costs, we now have the essential material to begin segmentation. Algorithm 1 gives the pseudo-code for the complete pipeline. The raw laser data is first processed into an augmented set of vertex V and a set of edges E. In line 2, we compute edge weights ω_α, ω_β and $\omega_{(u,v)}$, and in lines 4–6 we

Algorithm 1 Outline of the Point Cloud Clustering Algorithm

Input:

A frame of point cloud $P = (p_1, p_2, p_3, \ldots)$;

Output:

The segmentation result $P_{segment} = (s_1, s_2, s_3, \ldots)$.

1: Construct a undirected graph $G = (V, E)$ with the points as nodes V and the links between adjacent nodes as edges E.
2: Calculate the weights between adjacent points represented with ω_α, ω_β and $\omega_{(u,v)}$.
3: Start with a segmentation $S^0_{segment} = (s^0_1, s^0_2, s^0_3, \ldots, s^0_n)$, where each vertex is in its own component.
4: **for** $P_i \in P$ **do**
5: Let ω_α denote two adjacent vertices connect. If the weight of ω_α is less than the initial threshold $thres_\alpha$, merge the two points together as the ground point s_{ground} otherwise do nothing.
6: **end for**
7: **for** $P_i \in P$ and $P_i \notin s_{ground}$ **do**
8: Let ω_β and $\omega_{(u,v)}$ denote the weight between to adjacent points. If the two points belong different components and $|90° - \omega_\beta| < thres_\beta$, $\omega_{(u,v)} < thres_{(u,v)}$, merge the two components otherwise do nothing.
9: **end for**
10: **return** $P_{segment} = \cup_i s_i$

extract the ground points according to edged weights $thres_\alpha$. In lines 7–9, we traverse the edges and propose a union between the sets connected by the edges. Thus, we obtain the segmentation result. Typical clustering results using a 64-beam Velodyne scanner are shown in Figs. 7. Fig. 7 (a) shows the point cloud from the Velodyne, which is shown for illustration reasons only. The segmentation results between the ground and the objects are shown in Fig. 7 (b). The green points represent the ground and the red points represent the objects. The clustering result is shown in Fig.7 (c), where each object is surrounded by a bounding box.

C. Quadric Surface Fitting for Intra-Prediction

The intra-prediction method is inspired by the depth modeling modes (DMMs) in 3D-HEVC. The depth map coding in 3D-HEVC will be described firstly, followed by our proposed intra-prediction based on quadric surface fitting.

1) Depth Map Coding Method in 3D-HEVC: 3D-HEVC, as an extension of the HEVC standard, is specially designed for encoding multi-view video and depth data, which are primarily captured by RGB-D sensors. To better preserve edge details in depth video, 3D-HEVC adapts DMM as new prediction modes. The range image converted from a LiDAR has similar characteristics to the depth image captured by RGB-D sensor. They all have sharp edges and uniform areas. The difference is their measurement range and accuracy.

An RGB-D sensor has a measurement range of 4.5 meters, while a LiDAR can measure objects up to 100 meters away. Therefore, it requires more bits to represent the distance than in a depth image from an RGB-D sensor. Additionally, the

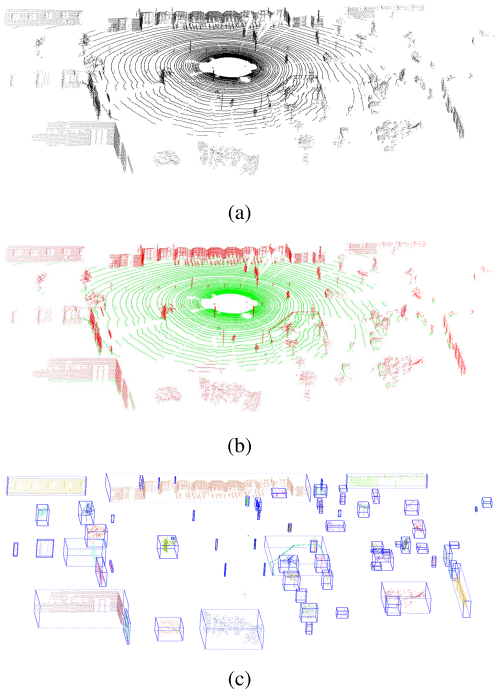


Fig. 7. Clustering results: (a) input point cloud; (b) extraction of the ground point, where the green points represent the ground, while red points represent the objects; (c) clustering result.

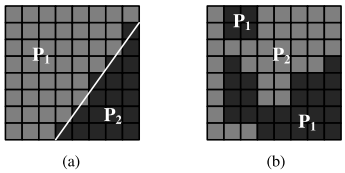


Fig. 8. Depth modeling modes: (a) wedgelet partitioning; (b) contour partitioning.

depth map captured by RGB-D sensor is dense, while the range image converted from LiDAR is sparse, especially for the points far from the LiDAR center. These reasons determine that we can not encode LiDAR data with a depth map coding method. However, we can learn how to eliminate spatial redundancy.

3D-HEVC exploits two novel DMMs. When using DMMs, the current block will be divided into two regions, P_1 and P_2, each of which will be represented by a constant partition value (CPV). The DMMs implement two segmentation strategies: wedge and contour segmentation. An example of wedge and contour partitioning is shown in Figure 8. For a wedge partition, as illustrated in Fig 8 (a), the depth prediction unit (PU) is split into two segments using a line defined by a starting point and an endpoint corresponding to the boundary sample. For contour segmentation, as illustrated in Fig 8 (b), the depth PU is segmented into arbitrary shapes using its collocated texture block information, and can be composed of multiple parts. The difference between the real data block $B_{true}(x, y)$ and the predicted data block $B_{predicted}(x, y)$ is calculated as the residual data, represented by $B_{residual}(x, y)$. Since the pixel value of the residual block is close to zero, the entropy of the

residual block is small. The remaining blocks can be encoded with very few bits compared to the original block. In addition, the split information will be stored in a binary matrix equal to the block size:

$$B_{residual}(x, y) = B_{true}(x, y) - B_{predicted}(x, y). \quad (4)$$

2) Intra-Prediction Based on Quadric Surface Fitting: Inspired by the prediction technique adopted in 3D-HEVC, we exploit a similar approach. After segmentation, the range image is represented as a set of components $P = \cup_i s_i$, as illustrated in Fig. 4 (d). Each component represents an object in 3D space. Similar to the DMMs in 3D-HEVC, for a component with few points, we use the average value to make the prediction. To further improve the coding efficiency, we also use the quadric surface to make prediction for the components with a large number of points. For given point cloud data $\{p_k\}_{k=1}^{n} \in s_i$, the least-squares function is defined as follows:

$$J = \frac{1}{n}\sum_{i=1}^{n} d(s, p_i)^2, \quad (5)$$

where $p_i = (x_i, y_i, z_i)^T$ represents a 3D point coordinate, and s denotes the parameter of the surface. Since the expression of the distance function directly affects the solution of nonlinear equations and the accuracy of the least squares method, we establish the parametric equation of the distance function of the quadric surface. Here, we define four conicoid models:

Plane model: the distance from a point p_i to a plane can be expressed as

$$d(s, p_i) = n p_i + \overline{d}, \quad (6)$$

where n denotes the normal vector of the plane, and $\overline{d}$ represents the vertical distance parameter from the origin of the coordinate to the plane.

Sphere model: the distance equation from a point p_i to a sphere can be expressed as

$$d(s, p_i) = ||c - p_i|| - r, \quad (7)$$

where c represents the center of the sphere, and r is the radius of the sphere.

Cylinder model: the distance from a point p_i to a sphere is defined as follows:

$$d(s, p_i) = \sqrt{||q_0 - p_i||^2 - (a \cdot (q_0 - p_i)^2)} - r, \quad (8)$$

where $q_0 = (k_1, k_2, k_3)^T$ represents the coordinates of a point on the axis of a cylinder, $a = (k_4, k_5, k_6)$ denotes the normal vector of the cylinder axis, and r is the cylinder radius k_7. As the point q_0 on the axis is not unique, it will affect the robustness of the algorithm. To make the obtained parameters independent, the distance between the point to the cylinder is re-parameterized. The point q_0 is redefined as the closest point to the origin on the axis of cylinders.

Cone model: the distance from a point p_i to a cone can be defined as follows:

$$\begin{aligned} d(s, p_i) &= |p_i B| \\ &= \sqrt{||c - p_i|| - n \cdot (c - p_i) \cdot cos\gamma - n \cdot (c - p_i) sin\gamma}, \end{aligned} \quad (9)$$

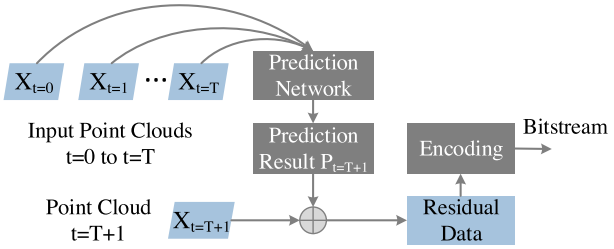


Fig. 9. Architecture of the inter-coding process. The grey blocks indicate processing steps and the blue blocks represent data.

where $c = (x_0, y_0, z_0)^T$ represents the apex of the conical surface, $n = (cos\alpha cos\beta, sin\alpha sin\beta, cos\alpha)$ denotes the unit vector in the direction of the cone axis (pointing to the apex), and γ is the cone's apex semi-angle. Thus, the geometric parameters of the conical surface are redefined as $(x_0, y_0, z_0, \alpha, \beta)$. The parameters are independent of each other.

After establishing the plane, spherical, cylindrical, and conical surface distance functions, the fitting calculation is performed. We use the coordinates of the point and its normal vector information to obtain the initial value estimation. After we obtain the initial values and distance equation, we use the Levenberg-Marquardt algorithm to get the optical surface parameters [31]. Using the fitted surface and LiDAR parameters, we calculate the virtual points and convert these points into a predicted range data. The difference between the real range data and the predicted range data is calculated as residual data for further processing.

D. Contour Map Coding Method

For each segment, the fitted plane parameters are saved in laser scanning order. To be able to reconstruct point clouds, we also need to save the contor map (Fig. 4 (f)). If one of the pixels in $p(x+1, y)$ or $p(x, y+1)$ belongs to a different segment, we set the pixel $p(x, y)$ to 1, and otherwise to 0. The extraction result of the contour map is shown in Fig. 4(f). After that, the boundary map is uniformly subdivided into 4×4 macroblocks, represented by 16-bit integer values.

$$V_{block} = \sum_{i=0}^{15} Contour(i) \cdot 2^i \quad (10)$$

where V_{block} represents the value assigned for the current block. $Coutor(i)$ is 1 if pixel i is on the boundry and 0 otherwise. Thus, we convert the contour map encoding into a 1D array encoding problem that can be encoded by any lossless compression scheme.

V. Inter-Prediction Technique Using Convolutional LSTM

When a car mounted with a Velodyne LiDAR is traveling in urban area, the time interval between two adjacent point clouds is very short, and the car only moves a short distance during this time. Adjacent frames in a point cloud sequence have a large number of similar structures, and there is a lot of redundancy in the time dimension. To eliminate temporal redundancy in point cloud sequences, we develop a prediction

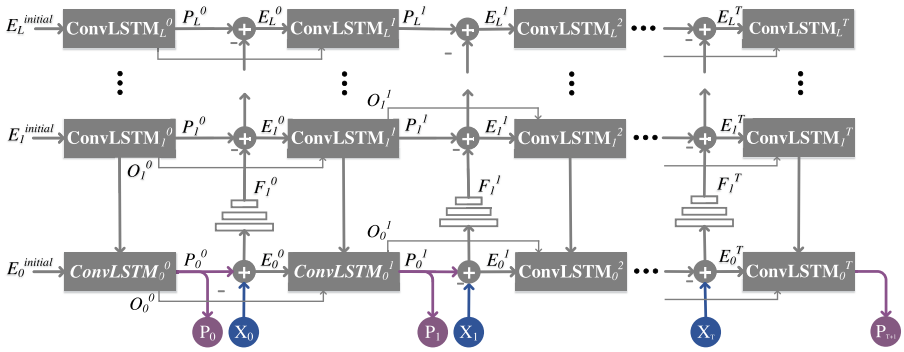


Fig. 10. Architecture of the point cloud prediction network.

neural network using convolutional LSTM, which is capable of inferring future point clouds according to the encoded frames. Fig. 9 gives the architecture the inter-coding process. The predition network obtains the encoded points clouds $X = \{X_{t=0}, X_{t=1}, \ldots, X_{t=T}\}$, and infers the next frame of point cloud $P_{t=T+1}$. The difference between the real point cloud $X_{t=T+1}$ and predicted result $P_{t=T+1}$ will be calculated, quantified and encoded as the inter-bitstream.

Deep learning algorithms have been widely used to solve supervised learning tasks. However, point cloud prediction, as unsupervised learning, remains a difficult challenge. Figure 10 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. $X = \{X_0, X_1, \ldots, X_T\}$ represents the input range images from $t = 0$ to T, while the $P = \{P_0, P_1, \ldots, P_T, P_{T+1}\}$ 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 2. The status update is performed through two processes: a top-down process in which the P_l^t state is calculated ad described in formula (17), and then a forward process is performed to calculate the error E_l^t, and higher-level targets F_l^t described in formulas (18) and (19). 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.

$$P_l^t = \begin{cases} ConvLSTM(E_L^{initial}) & t = 0, l = L \\ ConvLSTM(E_l^{initial}, O_{l+1}^t) & t = 0 \\ ConvLSTM(E_l^t, O_l^{t-1}) & l = L \\ ConvLSTM(E_l^t, O_l^{t-1}, O_{l+1}^t) & others \end{cases} \tag{11}$$

Algorithm 2 The Update Process of the Prediction Network

Input:

The input t frames: $I = \{X_0, X_1, \ldots, X_t\}$;

Output:

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

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**
      $P_L^t = ConvLSTM(E_L^{initial})$
    else if (t = 0) **then**
      $P_l^t = ConvLSTM(E_l^{initial}, O_{l+1}^t)$
    else if (l = L) **then**
      $P_L^t = ConvLSTM(E_l^t, O_l^{t-1})$
    else
      $P_l^t = ConvLSTM(E_l^t, O_l^{t-1}, O_{l+1}^t)$
    end if
  end for
  for $l = 0$ **to** $L - 1$ **do**
    if (l = 0) **then**
      $E_0^t = P_t - X_t$
      $F_0^t = Conv(E_0^t)$
    else
      $E_l^t = P_l^t - F_l^t$
      $F_l^t = Conv(E_l^t)$
    end if
  end for
end for

$$E_l^t = \begin{cases} P_t - X_t & t = 0 \\ P_l^t - F_l^t & others \end{cases} \tag{12}$$

$$F_l^t = \begin{cases} Conv(E_0^t) & t = 0 \\ Conv(E_l^t) & others \end{cases} \tag{13}$$

The model is trained to minimize the weighted sum of error cell activities. Explicitly, the training loss is formalized in equation (20), 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

TABLE I
EVALUATION OF NEXT-FRAME PREDICTIONS WITH KITTI DATASET

Point Cloud Scene	Campus	City	Road	Residential
MSE	0.52	0.72	0.81	0.86
SSIM	0.93	0.83	0.90	0.85

as follows.

$$Loss = \sum_{t=0}^{t=T} \sum_{l=0}^{l=L} \lambda_t \cdot \frac{\lambda_l}{l} \cdot ||E_l^t||. \tag{14}$$

We train the proposed prediction model using the KITTI dataset. We convert the point clouds to range images with resolution 64×2000 pixels. After that, the images are processed to be grayscale, with values normalized between 0 and 1. We use 5K point clouds for training and 600 for both validation and testing. The Adam method is used to train our model with learning rate 0.0001. We train our model with entropy optimization within 500 epochs. The batch size is set to 8, 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.

Quantitative assessment of generated models is a difficult and unresolved problem. We calculate the prediction errors based on mean square error (MSE) and structural similarity index measures (SSIM). The SSIM is designed to be more relevant to perceptual judgment, ranging from -1 to 1, with larger scores indicating greater similarity. Table I shows the results of four point cloud scenes: campus, city, road, and residential. The error between the real point cloud and predicted is calculated and perpared for further processing.

VI. EXPERIMENTAL RESULTS

A. Experimental Conditions and Evaluation Metric

To evaluate the point cloud compression performance, the main body of the proposed algorithm is implemented by C++ with some operations in the open source Point Cloud Library (PCL). The prediction network uses PyTorch 0.4.1 with the CUDA 8.0 and cuDNN 7.0 libraries. The experiments are performed on a PC with an Intel 2.2GHz i7 CPU and a single NVIDIA graphics card with 16GB memory. We use the public KITTI dataset to perform the experiments, using four scenes of point clouds: campus, city, road, and residential. We make comparisons with the available octree method [10], Google Draco [11], MPEG TMC13 [12], [32], and other recently proposed methods [33], [34]. The experiment tests 100 frames per sequence. The order of intra- and inter-frames is formatted as "IIIIIPPP…PPPIIIII…". Five intra-frames are encoded firstly, followed by fifteen inter-frames.

We adopt two metrics to quantitatively evaluate of the compression performance. The first is the compression rate, which represents the ratio between the compressed data size and the original size:

$$Ratio = \frac{Compressed_{size}}{Input_{size}} \times 100\%, \tag{15}$$

where $Ratio$ represents the compression ratio, and $Input_{size}$ and $Compressed_{size}$ represent the size of the point cloud data before and after compression, respectively.

The second is the point to point symmetric root mean square error (RMSE). 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.

$$\begin{aligned} P_{input} &= \{(p_i) : i = 0, \ldots, K-1\} \\ P_{decode} &= \{(p_i) : i = 0, \ldots, N-1\}. \end{aligned} \tag{16}$$

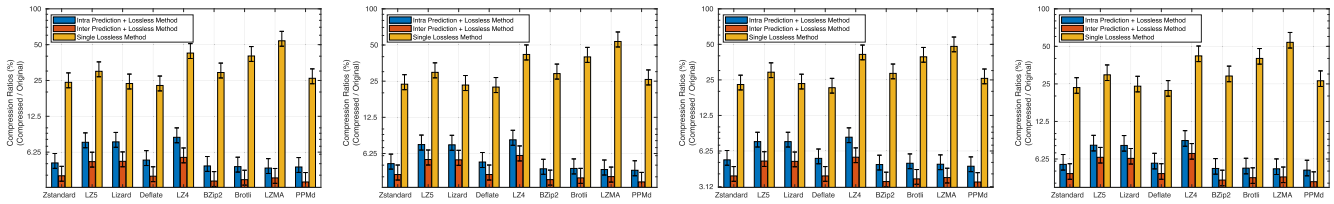
For each point in P_{input}, we take the distance to the nearest point in P_{decode}, represented as $p_{nn-decode}$. The $p_{nn-decode}$ is efficiently computed via a K-d tree in the L2 distance norm. The RMSE is defined as follows:

$$\begin{aligned} MSR(P_{input}, P_{decode}) &= \frac{1}{K} \sum_{p_l \in P_{input}} ||p_l - p_{nn-decode}||_2^2 \\ MSR(P_{decode}, P_{input}) &= \frac{1}{N} \sum_{p_l \in P_{decode}} ||p_l - p_{nn-input}||_2^2 \\ RMSE &= \sqrt{\frac{MSR(P_{input}, P_{decode}) + MSR(P_{decode}, P_{input})}{2}} \end{aligned} \tag{17}$$

B. Experimental Results

1) Intra- and Inter-Compression Ratio for a Single Point Cloud: We consider several compression schemes to encode the intra- or inter-predicted residual data, namely, Zstandard, LZ5, Lizard, Deflate, LZ4, Bzip2, Brotli, LZMA and PPMd. To make a comparision, the point cloud data is also directly encoded with these schemes without any pre-processing as an anchor. Figure 11 depicts the average intra- and inter- compression ratio for a single point cloud of four scenes. The intra-coding results using intra-prediction and different lossless compression algorithms are represented by blue bars, while the inter-coding result are represented by red bars. The coding results of point cloud data without using intra- or inter-prediction are represented by blue bars. It can be observed that both proposed intra-prediciton and inter-prediction methods paired with the lossless coding schemes yields better performance.

For intra-coding, the proposed intra-prediction method combined with the PPMd scheme obtains a lower compression ratio compared with the other coding schemes. Among the four scenes of point clouds, the compression performance of the residential scene is relatively poor. This is because the residential point cloud is complex, which results in more clusters and a more complex contour map. More bits are needed to encode the contour map and surface parameters for the clusters. The best compression ratio is obtained by the intra-predition paired with PPMd for the city scene point cloud, with 4.5%. The intra-coding results demonstrate that the



(a) (b) (c) (d)

Fig. 11. Intra- and inter- compression ratios for a single point cloud of four sences: (a) campus, (b) city, (c) road, (d) residential.

TABLE II
COMPRESSION RATE RESULTS COMPARED WITH THE OCTREE AND DRACO GOOGLE METHODS

Point Cloud Scene	Proposed method				Octree [10]			Google Draco [11]			
	Compression-accuracy				Distance Resolution			Compression-accuracy			
	$2mm$	$5mm$	$1cm$	$5cm$	$1mm^3$	$5mm^3$	$1cm^3$	$1mm$	$5mm$	$1cm$	$5cm$
Campus	3.94	3.14	2.52	1.48	21.27	8.05	5.75	11.87	7.75	5.47	2.08
City	4.06	3.22	2.64	1.46	23.98	10.76	8.40	12.52	8.38	6.49	3.39
Road	3.85	3.11	2.61	1.56	23.56	10.35	7.99	12.31	8.35	6.59	3.09
Residential	4.55	3.74	3.16	1.83	22.94	9.72	7.37	12.66	8.59	6.33	3.06
Average	**4.10**	**3.30**	**2.73**	**1.58**	**20.23**	**9.72**	**7.29**	**12.34**	**8.27**	**6.22**	**3.87**

intra-prediction technique can effectively remove the spatial redundancy of the point cloud data.

For inter-coding, the inter-frame prediction technology combined with the Bzip2 coding scheme achieves the optimal compression performance for the campus scene point cloud, with a compression ratio of 3.57%. Compression performances of the different scenarios do not differ much. Compared with the intra-coding method, the performance of the inter-coding is better. The inter-coding results demonstrate that the proposed prediction network is capable of removing the temporal redundancy of point cloud sequences by inferring future point cloud data.

2) Comparison With Octree and Google Draco: Table II describes the compression ratio results of the proposed method compared to the well-known octree method [10] and the recent proposed Goolge-Draco method [11]. Considering that the two algorithms can not directly encode point cloud sequences, we assemble point cloud sequences into a point cloud map by transforming each single point cloud into a global coordinate system. The coding accuracy of the octree coding technique is set to 1 cubic millimeter, 5 cubic millimeters, and 1 cubic centimeter. In our experiments, we choose the LZMA scheme as a representative of the lossless compression algorithm. From Table II, it can be seen that the algorithm obtains a smaller compression ratio than the octree coding technique.

DRACO is an open source library developed by Google that uses k-d tree data structures to quantify and organize points in 3D space. There are two custom parameters in the DRACO algorithm, namely, the quantization bit (QB) and the compression level (CL). In our experiments, the quantization bits are set to 17, 15, 14, and 11, corresponding to 1 mm, 5 mm, 1 cm, and 5cm accuracy, respectively. In addition, we set $cl = 10$ to get the highest compression ratio. The experimental results show that the compression scheme achieves a smaller compression ratio than the DRACO method.

3) RMSE-bpp Curves Performance: Figure 12 illustrates the performance of the proposed method compared with the Draco [11], TMC13 [32] and Tu methods [33], [34]. Comparison results are given by RMSE-bpp curves, which reflect the relationship between the RMSE and bits per point. TMC13 is an emerging point cloud compression standard lately released at the 125th MPEG meeting. We experiment on four scenes of point clouds: campus, city, road and residential. As Tu *et al.* did not publish their coding performance for the campus and city point clouds, for campus and city scenes, we only compare our method with the Draco and TMC13 methods. 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 Google Draco [11], TMC13 [32], U-net-based method and Tu *et al.*'s methods under the same RMSE reconstruction quality.

In [33], Tu *et al.* proposed a continuous point cloud data compression algorithm using SLAM-based prediction. They used the SLAM-based method to reconstruct the 3D environment [35], and make predictions according to the 3D environment and LiDAR parameters. In [34], Tu *et al.* presented a real-time LiDAR point cloud streams algorithm. They defined some frames as reference frames, and used U-net to interpolate the remaining frames [36]. In contrast with Tu *et al.*'s methods, our proposed prediction network can use the temporal characteristics of multi-frame point clouds to infer future frames. Also, the proposed model is a multi-layered recurrent neural network structure, where the error is passed from top to bottom to get more accurate prediction results. Moreover, the spatial redundancy of intra-frames is also removed by clustering and quadric surface fitting. Different from Tu *et al.*'s methods, TMC13 is an MPEG standard for point cloud compressions, which can be used for various types of point clouds. It is not specifically designed for Velodyne point clouds [37].

4) Contribution of Each Step in Intra- or Inter-Coding Process: To evaluate the contribution of each step in the encoding process, we record the change in the size of the point cloud data during the compression process. Table III illustrates

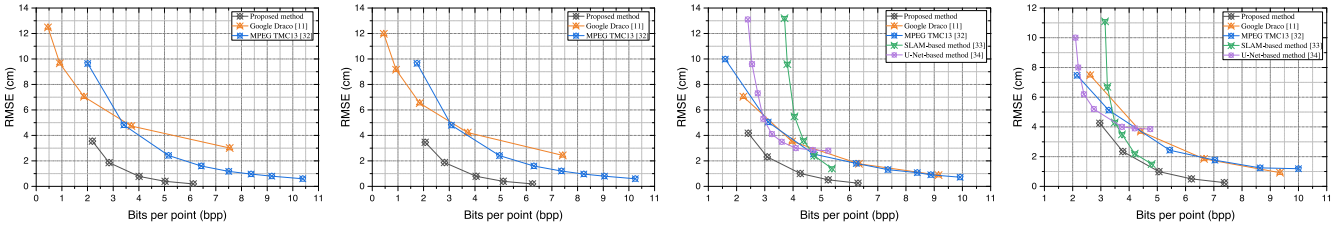


Fig. 12. RMSE-bpp curves performance of our method in comparision with Google Draco [11], TMC13 [32] and Tu *et al.*'s methods [33], [34]: (a) campus scene, (b) city scene, (c) road scene(d) residential scene. (Best viewed by zooming in.)

TABLE III
AVERAGE CHANGES IN DATA VOLUMES DURING INTRA- OR INTER- CODING (KB)

Scene	Original	Convertion	Intra/Inter Prediction	Coding
Campus	3134	835	533 / 373	150 / 112
City	3076	837	515 / 385	144 / 117
Road	3254	822	531 / 381	156 / 112
Residential	3147	823	539 / 427	166 / 134

TABLE IV
AVERAGE CODING TIME OF THE INTRA-CODING TECHNIQUE

Coding Method	Clustering	Intra-Prediction	Coding	Total
Intra & PPMd	0.13	1.59	0.23	1.95

the average change in the amount of data during the intra- and inter- coding processes. The last contribution comes from the coding part. The intra or inter residual data is quantified firstly, and encoded by lossless schemes. By adjusting the quantization parameters, the compression accuracy can be controlled. Here, we use the BZip2 method with millimeter accuracy. The steps of conversion, intra or inter prediction, and residual data encoding are all important techniques for the proposed point cloud sequence compression algorithm. Their contributions are not simply added together, but their effects are multiplied. Therefore, even a small improvement in a single step can make a big difference.

5) Speed Performance: The proposed intra-coding technique includes three steps: range image conversion, intra-prediction and residual data encoding, while the inter-coding technique includes range image conversion, inter-prediction and residual data encoding. We chose 50 frames to evaluate the speed performance of the proposed method. Tables IV and V give the average coding time of each step for intra-coding and inter-coding processes, respectively. It can be seen that the total coding time is 1.95s for intra-coding and the 0.77s for inter-coding.

Currently, the algorithm cannot be run in real-time, but the algorithm framework can be used offline. After we have collected the LiDAR data, we can use this algorithm to encode them to reduce the storage space. After compression, when transmitting these data to others through the internet, the bandwidth will also be reduced. In future work, we will continue to optimize the point cloud coding architecture to improve the coding efficiency and reduce complexity. We will try to find a balance between the algorithm complexity and the compression rate.

TABLE V
AVERAGE CODING TIME OF THE INTER-CODING TECHNIQUE

Coding Method	Conversion	Inter-Prediction	Coding	Total
Inter & PPMd	0.11	0.42	0.24	0.77

VII. CONCLUSION

In this article, we propose a novel coding architecture for multi-line LiDAR point cloud sequences. To remove the spatial redundancy, an intra-prediction technique is developed based on clustering and quadric surface fitting. To remove the temporal redundancy, we develop a prediction network using convolutional LSTM cells. The network can infer future inter-frames according to the encoded intra-frames. Experimental results show that both the intra and inter coding achieve a high compression rate. The proposed algorithm also outperforms octree [10], Google Draco [11], MPEG TMC13 [32] and other recently proposed methods [33], [34].

The proposed method is specially desinged for LiDAR point cloud sequences captured by line-laser scanners for autonomous vehicles. Intra-frame prediction is also applicable to organized dense point clouds [38], [39]. However, the proposed method can not apply to disordered point cloud compression, such as 3D human body sequences [40]. Additionally, the high complexity of the algorithm can not satisfy real-time applications at present. Future work will focus on improving the applicability and reducing the complexity of the proposed algorithm.

REFERENCES

[1] H. Ye, Y. Chen, and M. Liu, "Tightly coupled 3D Lidar inertial odometry and mapping," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 3144–3150.
[2] M. Liu, "Robotic online path planning on point cloud," *IEEE Trans. Cybern.*, vol. 46, no. 5, pp. 1217–1228, May 2016.
[3] P. Yun, L. Tai, Y. Wang, C. Liu, and M. Liu, "Focal loss in 3D object detection," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 1263–1270, Apr. 2019.
[4] J. Elseberg, D. Borrmann, and A. Nüchter, "One billion points in the cloud – an octree for efficient processing of 3D laser scans," *ISPRS J. Photogramm. Remote Sens.*, vol. 76, pp. 76–88, Feb. 2013.
[5] J. Kammerl, N. Blodow, R. B. Rusu, S. Gedikli, M. Beetz, and E. Steinbach, "Real-time compression of point cloud streams," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2012, pp. 778–785.

[6] C. Tu, E. Takeuchi, C. Miyajima, and K. Takeda, "Compressing continuous point cloud data using image compression methods," in *Proc. IEEE 19th Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2016, pp. 1712–1719.

[7] M. Wang, K. N. Ngan, and H. Li, "An efficient frame-content based intra frame rate control for high efficiency video coding," *IEEE Signal Process. Lett.*, vol. 22, no. 7, pp. 896–900, Jul. 2015.

[8] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3354–3361.

[9] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, Sep. 2013.

[10] D. Meagher, "Geometric modeling using octree encoding," *Comput. Graph. Image Process.*, vol. 19, no. 2, pp. 129–147, Jun. 1982.

[11] Google. (2018). *Draco: 3D Data Compression*. [Online]. Available: https://github.com/google/draco

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

[13] X. Liu, Y. Wang, Q. Hu, and D. Yu, "A scan-line-based data compression approach for point clouds: Lossless and effective," in *Proc. 4th Int. Workshop Earth Observ. Remote Sens. Appl. (EORSA)*, Jul. 2016, pp. 270–274.

[14] H. Houshiar and A. Nuchter, "3D point cloud compression using conventional image compression for efficient data transmission," in *Proc. 25th Int. Conf. Inf., Commun. Autom. Technol. (ICAT)*, Oct. 2015, pp. 1–8.

[15] J.-K. Ahn, K.-Y. Lee, J.-Y. Sim, and C.-S. Kim, "Large-scale 3D point cloud compression using adaptive radial distance prediction in hybrid coordinate domains," *IEEE J. Sel. Topics Signal Process.*, vol. 9, no. 3, pp. 422–434, Apr. 2015.

[16] P. Zanuttigh and G. M. Cortelazzo, "Compression of depth information for 3D rendering," in *Proc. 3DTV Conf., True Vis. Capture, Transmiss. Display 3D Video*, May 2009, pp. 1–4.

[17] V. Morell, S. Orts, M. Cazorla, and J. Garcia-Rodriguez, "Geometric 3D point cloud compression," *Pattern Recognit. Lett.*, vol. 50, pp. 55–62, Dec. 2014.

[18] C. Yang, Z. Wang, W. He, and Z. Li, "Development of a fast transmission method for 3D point cloud," *Multimedia Tools Appl.*, vol. 77, no. 19, pp. 25369–25387, Oct. 2018.

[19] X. Wang, Y. A. Sekercioglu, T. Drummond, E. Natalizio, I. Fantoni, and V. Fremont, "Fast depth video compression for mobile RGB-D sensors," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 4, pp. 673–686, Apr. 2016.

[20] R. A. Cohen *et al.*, "Compression of 3-D point clouds using hierarchical patch fitting," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 4033–4037.

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

[22] L. Wang, L. Wang, Y. Luo, and M. Liu, "Point-cloud compression using data independent method—A 3D discrete cosine transform approach," in *Proc. IEEE Int. Conf. Inf. Autom. (ICIA)*, Jul. 2017, pp. 1–6.

[23] Y. Fan, Y. Huang, and J. Peng, "Point cloud compression based on hierarchical point clustering," in *Proc. Asia–Pacific Signal Inf. Process. Assoc. Annu. Summit Conf.*, Oct. 2013, pp. 1–7.

[24] H. Q. Nguyen, P. A. Chou, and Y. Chen, "Compression of human body sequences using graph wavelet filter banks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2014, pp. 6152–6156.

[25] D. Thanou, P. A. Chou, and P. Frossard, "Graph-based motion estimation and compensation for dynamic 3D point cloud compression," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2015, pp. 3235–3239.

[26] R. L. de Queiroz and P. A. Chou, "Motion-compensated compression of point cloud video," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 1417–1421.

[27] X. Zhang *et al.*, "Low-rank-based nonlocal adaptive loop filter for high-efficiency video compression," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 10, pp. 2177–2188, Oct. 2017.

[28] W. Lotter, G. Kreiman, and D. Cox, "Deep predictive coding networks for video prediction and unsupervised learning," 2016, *arXiv:1605.08104*. [Online]. Available: http://arxiv.org/abs/1605.08104

[29] R. B. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 1–4.

[30] B. Matejek, D. Haehn, F. Lekschas, M. Mitzenmacher, and H. Pfister, "Compresso: Efficient compression of segmentation data for connectomics," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent.* Cham, Switzerland: Springer, 2017, pp. 781–788.

[31] J. J. Moré, "The Levenberg-Marquardt algorithm: Implementation and theory," in *Numerical Analysis*. Cham, Switzerland: Springer, 1978, pp. 105–116.

[32] *PCC WD G-PCC (Geometry-Based PCC)*, document ISO/IEC JTC1/SC29/WG11 MPEG. N 17771, 3D Graphics, Jul. 2018.

[33] C. Tu, E. Takeuchi, C. Miyajima, and K. Takeda, "Continuous point cloud data compression using SLAM based prediction," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2017, pp. 1744–1751.

[34] C. Tu, E. Takeuchi, A. Carballo, and K. Takeda, "Real-time streaming point cloud compression for 3D LiDAR sensor using U-Net," *IEEE Access*, vol. 7, pp. 113616–113625, 2019.

[35] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based SLAM," *IEEE Intell. Transp. Syst. Mag.*, vol. 2, no. 4, pp. 31–43, 2010.

[36] H. Jiang, D. Sun, V. Jampani, M.-H. Yang, E. Learned-Miller, and J. Kautz, "Super SloMo: High quality estimation of multiple intermediate frames for video interpolation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9000–9008.

[37] R. Halterman and M. Bruch, "Velodyne HDL-64E lidar for unmanned surface vehicle obstacle detection," *Proc. SPIE*, vol. 7692, May 2010, Art. no. 76920D.

[38] P. Huang, M. Cheng, Y. Chen, H. Luo, C. Wang, and J. Li, "Traffic sign occlusion detection using mobile laser scanning point clouds," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 9, pp. 2364–2376, Sep. 2017.

[39] H. Guan, J. Li, Y. Yu, M. Chapman, and C. Wang, "Automated road information extraction from mobile laser scanning data," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 1, pp. 194–205, Feb. 2015.

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

Xuebin Sun received the bachelor's degree from the Tianjin University of Technology in 2011 and the master's and Ph.D. degrees from Tianjin University in 2014 and 2018, respectively. He is currently a Research Fellow with the Surgical Robotics Laboratory, Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hong Kong. His research interests include video coding optimization, digital image processing, deep learning, and point cloud processing algorithms.

Sukai Wang (Graduate Student Member, IEEE) received the bachelor's degree in measurement control and instrument science from Zhejiang University in 2018. He is currently pursuing the Ph.D. degree with the Department of Electronic and Computer Engineering, Robotics Institute, The Hong Kong University of Science and Technology (HKUST), Hong Kong. His research interests include navigation, autonomous car, and deep learning.

Ming Liu (Senior Member, IEEE) received the B.A. degree in automation from Tongji University in 2005 and the Ph.D. degree from the Department of Mechanical and Process Engineering, ETH Zürich, in 2013, supervised by Prof Roland Siegwart. He is currently affiliated with the ECE Department, CSE Department, and Robotics Institute, Hong Kong University of Science and Technology. His research interests include dynamic environment modeling, deep-learning for robotics, 3D mapping, machine learning, and visual control. He received twice the innovation contest Chunhui Cup Winning Award in 2012 and 2013 and the Wu Weijun AI Award in 2016. He was the Program Chair of the IEEE-RCAR 2016 and the Program Chair of International Robotics Conference in Foshan 2017. He is the Conference Chair of ICVS 2017.