



# A Novel Point Cloud Compression Algorithm Based on Clustering

Xuebin Sun , Han Ma, Yuxiang Sun , and Ming Liu 

**Abstract**—Due to the enormous volume of point cloud data, transmitting and storing the data requires large bandwidth and storage space. It could be a critical bottleneck, especially in tasks such as autonomous driving. In this letter, we propose a novel point cloud compression algorithm based on clustering. The proposed scheme starts with a range image-based segmentation step, which segments the three-dimensional (3-D) range data into ground and main objects. Then, it introduces a novel prediction method according to the segmented regions' shape. This prediction method is inspired by the depth modeling modes used in 3-D high-efficiency video coding for depth map coding. Finally, the few prediction residual is efficiently compressed with several lossless or lossy data compression techniques. Experimental results show that the proposed algorithm can largely eliminate the spatial redundant information of the point cloud data. The lossless compression scheme reaches a compression ratio of nearly 5%, which means that the point cloud is compressed to 5% of its original size without any distance distortion. Compared with other methods, the proposed compression algorithm also shows better performance.

**Index Terms**—Range Sensing, automation technologies for smart cities, SLAM.

## I. INTRODUCTION

### A. Motivation

IN RECENT years, thanks to the increasing capabilities of 3D data acquisition in computer vision and autonomous driving technology, there has been an extensive proliferation of 3D point cloud applications in autonomous driving technologies such as simultaneous localization and mapping (SLAM) [1], autonomous navigation [2], object detection [3] and tracking [4]. A point cloud is often composed of a considerable number of points that capture highly detailed geometric information, along with one or more attributes, such as color or normal. Modern 3D laser sensors, for instance, the Velodyne HDL64, are capable of measuring more than 120 thousand points per frame, which imposes a huge burden for storage and transmission with current technology. Therefore, compression of point

data has become an essential task for the further development of autonomous driving.

A point cloud is often composed of a large number of points over a large area. Thus, it is difficult to remove redundancy with the existing methods. Point cloud data from Velodyne HDL64 is structured, and can be converted into range images. However, it is not ideal to directly use an image coding technique as a compressing method. For one reason, the traditional image or video compression algorithms (i.e., JPEG2000, JPEG\_LS, High Efficiency Video Coding (HEVC)), can encode only integer pixel values, which are not suitable for Light Detection And Ranging (LIDAR) data recoding in floating-point numbers. Additionally, each pixel in the image represents the color value, whereas the pixel in the range image denotes the distance from the object to the LIDAR. It is inefficient to encode the range image using the image-based prediction methods. Considering the loss of information, the Voxel grid or Octree-based methods are not good choices for autonomous driving. As in object detection or obstacle avoidance algorithm, even a small loss of information may result in an error.

To avoid distortion and information loss, we propose a lossless compression scheme based on point cloud clustering. Besides that, lossy compression methods are also explored. The proposed method benefits from a novel prediction technique, which takes the correlation of the distance information of points into consideration to remove spatial redundancies rather than using the image prediction method directly. Compared with image-based point cloud compression methods, the proposed approach achieves better performance in terms of the compression ratio.

### B. Contributions

In this letter, we propose a novel point cloud compression framework for the structured LIDAR point cloud data. Firstly, we convert the structured point cloud data into range image, which can save nearly three-fourths of the storage space. Secondly, inspired by the DMM technique adopted in 3D-HEVC [25], we use a clustering method to remove the spatial redundancy of the point cloud data. Thirdly, we use traditional lossless or lossy schemes to encode the residual data. Compared with the octree or image-based point cloud compression methods, our proposed method utilizes the spatial structure of point cloud data and removes the spatial redundancy. As a result, the proposed method obtains an impressive compression performance.

Manuscript received October 7, 2018; accepted January 31, 2019. Date of publication February 21, 2019; date of current version March 5, 2019. This letter was recommended for publication by Associate Editor G. Grisetti and Editor C. Stachniss upon evaluation of the reviewers' comments. This work was supported in part by the National Natural Science Foundation of China under Grant U1713211 and in part by the Research Grant Council of Hong Kong SAR Government, China, under Project No. 11210017 and No. 21202816. The work of M. Liu was supported by the HKUST Project IGN16EG12. (Corresponding author: Ming Liu.)

The authors are with the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Hong Kong (e-mail: eesxb@ust.hk; hmaad@connect.ust.hk; eeyxsun@ust.hk; eelium@ust.hk).

Digital Object Identifier 10.1109/LRA.2019.2900747

### C. Organization

The rest of this letter is organized as follows. Section II gives an outline of related work. Section III presents an overview of the point cloud coding framework. Then, a detailed description of the proposed method is brought in Section IV. Section V is devoted to the experimental results and comparison with other novel methods. Finally, the conclusion and future work are given in the last section.

## II. RELATED WORK

Over the past decade, many algorithms for point cloud compression have been proposed. According to the characteristics of point clouds, these compression algorithms can be roughly classified into two categories: structured and unstructured point cloud compression.

### A. Structured Point Cloud Compression

1) *LIDAR Point Cloud Data Compression*: Ahn *et al.* [5] focused on efficient geometry compression of large-scale 3D point cloud data generated by a terrestrial laser scanner. The point cloud data are mapped into range images coded by an adaptive radial distance prediction method. Their technique achieves better compression performance compared with conventional image or video coding algorithms. Tu *et al.* [6] explored compression of continuous point cloud data using image compression methods. They convert the raw point cloud data into range images, and use image or video coding algorithms to reduce the volume of the data. As the image-based point cloud method is unable to utilize the 3D characteristics of point clouds, Tu *et al.* [7] proposed a point cloud data compression method using SLAM-based prediction. By mapping the point cloud onto panoramic images, Houshiar *et al.* [8] exploited the use of conventional image-based compression methods for 3D point clouds.

2) *RGB-D Raw Data Compression*: Wang *et al.* [9] proposed an efficient compression method for RGB-D data, which combines ego-motion estimation, 3D image warping techniques, and a lossless coding scheme to efficiently remove the redundancy existing in depth images. Considering the virtual view distortion, Yang *et al.* [10] presented a region-based Quantization Parameter (QP) adjustment method to reduce the depth map coding complexity. Morell *et al.* [11] proposed a geometric 3D point cloud lossy compression system, in which the Random Sample Consensus (RANSAC) method is used to extract planes from a point cloud. The points in each scene plane are represented as a Delaunay triangulation and a set of point/area information. Compared with the Voxelgrid and Octree methods, the proposed scheme has similar behavior, but lower error in the reconstruction phase.

### B. Unstructured Point Cloud Compression

Tree-based methods, which can be lossless, are popular methods for compressing 3D data, like point clouds. Elseberg *et al.* [12] proposed an octree structure for efficient processing of 3D laser scans, which can be further used for file format exchange, fast point cloud visualization, and 3D scan matching. Kammerl

*et al.* [13] proposed a technique to compare the octree data structure difference of consecutive point clouds, by which the temporal redundancies of point cloud streams can be detected and removed. Zhang *et al.* [14] decomposed a 3D color point cloud into many homogeneous blocks, and calculated its fitting plane parameters using the RANSAC method for every subset [28]. Then, a 2D Discrete Cosine Transform (DCT) transform is performed to compress the color information of each grid. Golla *et al.* [15] propose a real-time compression method for point cloud data based on local 2D parameterizations of surface point cloud data. They use a pre-processing method to split data into height map and use standard image coding methods to compress local details. Gumhold *et al.* [16] propose a spanning prediction tree technique for point-cloud compression, in which they construct a spanning tree over the vertices, followed by the encoding of their corresponding prediction error. Schnabel *et al.* [17] proposed a progressive compression method for point sampled models based on an octree decomposition of space.

Some point cloud compression methods are specially designed for immersive 3D human body compression. Thanou *et al.* [18] designed a compression framework for a dynamic 3D point cloud sequence, where they introduce a novel approach for motion estimation and compensation for geometry and color information. Zhang *et al.* [19] presented a graph-based coding method to compress attributes on 3D point clouds, which adopts graph transformation to decorrelate the color signal. The similar approach in Nguyen *et al.* [20] uses graph wavelet filter banks to compress moving human body sequences. Their scheme outperforms traditional methods such as HEVC. Mekuria *et al.* [21] introduced a generic compression method for real-time 3D tele-immersive video, which is suitable for mixed reality applications. In their works, the octree structure is adopted in intra frames, and a prediction method is performed for inter coding by portioning the octree voxel space into macroblocks.

While the aforementioned approaches can significantly reduce the point cloud data size with ignorable distortions, few of them aims at compressing LIDAR point cloud data in autonomous driving. With its characteristics of enormous volume, uneven distribution, it is difficult to efficiently compress LIDAR data using these algorithms. In this study, we propose an efficient point cloud data compression approach, especially for LIDAR point cloud data.

## III. OVERVIEW OF POINT CLOUD CODING FRAMEWORK

In the letter, we address the problem of compression of point cloud data from 3D LIDAR used in the autonomous driving systems. The architecture of the point cloud compression scheme is presented in Fig. 1, where the blue parallelograms represent processing steps and the ochre parallelograms indicate data. In particular, the processed data, which need to be further compressed by lossless or lossy coding methods, are marked with an asterisk.

### A. Outlier Removal Filter

In practice, LIDAR sensors such as the Velodyne HDL-64 produce a large number of outliers during the range measurement. The outlier points tend to reduce the efficiency of the

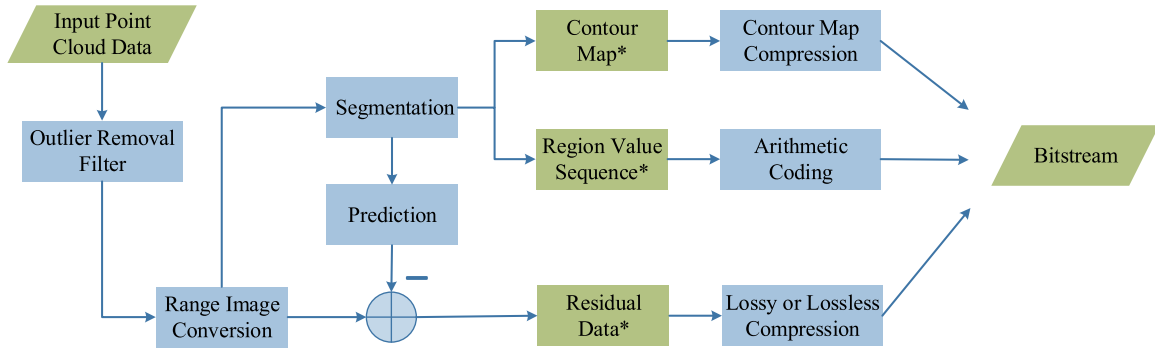


Fig. 1. Architecture of the point cloud compression scheme.

compression. Therefore, an efficient Radius Outlier Removal filter is utilized [29], which will remove the points in the input cloud that do not have at least a certain number of neighbors within a defined range.

### B. Range Image Conversion

Our experiment is performed with the KITTI dataset [22], which consists of several specific scenarios, such as road, city, residential and campus scenes. The point cloud data are captured by the Velodyne HDL-64E rotating 3D laser scanner, which covers  $26.8^\circ$  vertical and  $360^\circ$  horizontal fields of view, with 64 beams and  $0.18^\circ$  angular resolution. As the raw data are obtained by measuring the radial distances from the LIDAR center to the object, the data are structured and can be converted to range images.

### C. Segmentation

There is great redundancy information in the points belonging to the same object. Thus, an efficient clustering method is proposed. To avoid complex computation in 3D space, the clustering method is performed on the range images to segment 3D range data into different objects.

### D. Prediction

By using RANSAC [28], the points belonging to the ground are fitted by a 3D plane, and the difference between them is calculated. Moreover, similar to the DMM technique adopted in 3D-HEVC [25], we use the average depth value of each cluster to replace their real values. By using this method, we can obtain the predictive image. The residual data between the predicted and real range image need to be further processed.

### E. Contour Map Coding

The segmentation is partly stored as the contour image, consisting largely of low-frequency regions with structured boundaries. To remove redundancy across border regions, we use sliding windows to encode the contour information with integers [26].

### F. Arithmetic Coding

The values of each region are organized in laser scanning order and compressed by the arithmetic coding scheme.

### G. Lossy or Lossless Compression

In order to maximize the efficiency of the compression method, we explore both lossless and lossy methods to encode the residual data [27].

## IV. PROPOSED ALGORITHM

The proposed efficient point cloud compression strategy mainly consists of four techniques, range image conversion, point cloud segmentation, prediction, and coding method. Each technique will be detailed in this section. Figure 2 gives the overall workflow of the proposed methodology. The first step is the conversion of point clouds into range images. In the second step, a clustering process is executed based on the range image to segment the point cloud into road and main objects. The next step of the algorithm is the prediction of the range image based on the segmentation result. Finally, the contour map and residuals between the prediction and the real range image are computed and coded by either lossless or lossy coding methods.

### A. Range Image Conversion

Most LIDAR sensors provide raw data in the form of range distance per laser coupled with a timestamp and an orientation of the beam, which provides a chance to translate the data directly into a range image. The numbers of rows and columns are determined by the number of lines and horizontal resolution of the laser sensor, respectively. As the point cloud data of the LIDAR are organized, we losslessly project the 3D point cloud onto a spherical image by calculating the Euclidean distance per point. The results are shown in Fig. 2(a) and (b).

### B. Point Cloud Segmentation

Due to the high requirement of segmentation accuracy and timeliness, we use the segmentation approach from [23] and [24] without modification. The segmentation method is based on the range image. With the aim to extract the ground, we replace each column  $c$  of the range image  $R$  with a series of angles  $\alpha_{r-1,c}$ . Each of these angles represents the angle between two adjacent



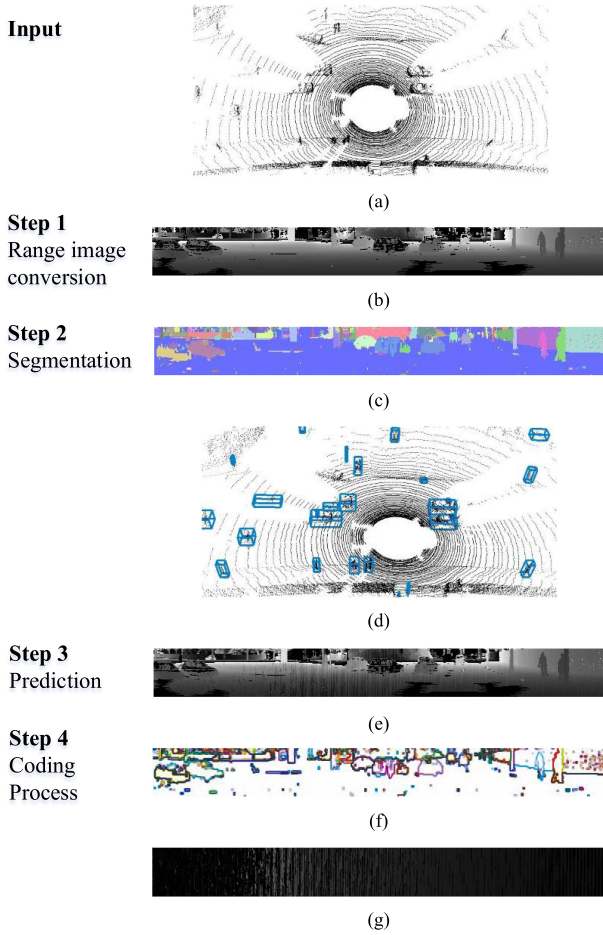


Fig. 2. The overall workflow of the proposed methodology: (a) The original point cloud; (b) Range image; (c) Segmentation result in range image; (d) Segmentation result in point cloud; (e) Prediction map; (f) Contours map; (g) Residual data.

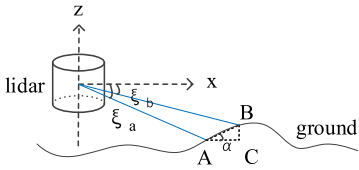


Fig. 3. Schematic diagram of the ground extraction algorithm.

points A and B and the horizontal surfaces, as illustrated in Fig. 3.

Point A and B are derived from two neighboring rows  $r - 1$  and  $r$  of the range image, represented by  $R_{r-1,c}$  and  $R_{r,c}$ , respectively. With a priori angle knowledge of vertically consecutive individual laser beams, the angle  $\alpha$  can be calculated using the following trigonometric rules:

$$\begin{aligned} \alpha &= \arctan(|BC'|, |AC'|) = \arctan(\Delta z, \Delta x) \\ \Delta z &= |R_{r-1,c} \sin \zeta_\alpha - R_{r,c} \sin \zeta_\beta| \\ \Delta x &= |R_{r-1,c} \cos \zeta_\alpha - R_{r,c} \cos \zeta_\beta|, \end{aligned} \quad (1)$$

where  $\zeta_\alpha$  and  $\zeta_\beta$  represent vertical angles of the laser beams corresponding to rows  $r - 1$  and  $r$ , respectively. If point A and

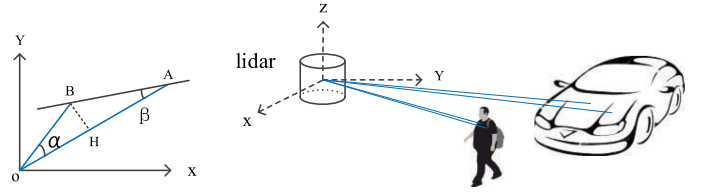


Fig. 4. Schematic diagram of the clustering method.

B belong to the ground, the value of  $\alpha$  will be small. A threshold of  $\alpha$  is defined to extract the ground.

After removing the ground from the point cloud, clustering is applied. Fig. 4 depicts the schematic diagram of the clustering method. Point A and B represent two random points measured from a LIDAR sensor located at the ordinate origin with the illustrated laser beams OA and OB.  $\beta$  represents the angle between the laser beam and the line connecting point A and point B. Practically, the angle  $\beta$  provides valuable information to judge whether two points belong to the same object or not.  $\beta$  is defined as follows:

$$\beta = \arctan \frac{|BH|}{|HA|} = \arctan \frac{d_2 \sin \alpha}{d_1 - d_2 \cos \alpha}, \quad (2)$$

where  $d_1$  and  $d_2$  represent the distance of OA and OB, respectively. The angle  $\alpha$  donates the angle between the beams, which can be obtained from the datasheet of the scanner.

The angle  $\beta$  always takes larger values for the points belonging to the same object. On the other hand, it takes small values when the depth difference between neighboring points given on the range image is substantially larger than the distance from one point to the other laser line. This phenomenon allows us to define a threshold  $\theta$  for the angle  $\beta$ , which plays the role of separating any two points into different clusters or merging them into one. If  $\beta$  is smaller than the threshold  $\theta$ , the two points are classified into different segments. Otherwise, the points are considered as belonging to the same object. Fig. 2(c) and (d) show the clustering results of the method.

Through the above steps, we not only extract the ground from the point cloud but we also classify the remaining points into different categories, which paves the way for further prediction. The next section details how to generate the prediction map according to the clustering results.

### C. Prediction

1) *Region Prediction Inspired by DMM*: 3D-HEVC is an extension of the HEVC standard, which is used for encoding multi-view video and depth data, captured by RGB-D sensors. The range image of LIDAR is similar to the depth image in 3D video. However, compared with the depth image, points in range image need more bits to represent the distance.

For the depth map, the 3D-HEVC encoder uses the DMM as a new additional intra coding tools. The DMM consists of two splitting methods, namely, wedgelet and contour splitting. The prediction block is divided into two independent areas by the splitting method, as illustrated in Fig. 5. For a wedgelet segmentation, the two areas are split by a straight line, while

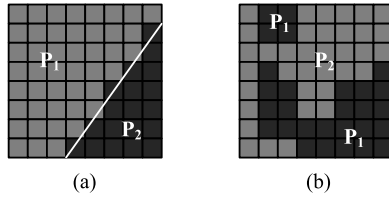


Fig. 5. Depth Modeling Modes (a) wedgelet partitioning (b) contour partitioning.

contour splitting uses an arbitrary shaped curve to separate the block. Each segmented region is predicted by using a Constant Partition Value (CPV). Block subtraction between the predicted block  $B_{predict}(x, y)$  and the true block  $B_{true}(x, y)$  is used to obtain the residual block  $B_{residual}(x, y)$ :

$$B_{residual}(x, y) = B_{true}(x, y) - B_{predict}(x, y) \quad (3)$$

As the pixel values of the residual block are all nearly zero, the entropy of the block is smaller. Compared with the true block, we can use few bits to encode the residual block. Moreover, split information will be stored in a binary matrix equal to the block size.

Inspired by the DMM technique adopted in 3D-HEVC for the depth maps coding [23], we adopt a similar approach. As depicted in Fig. 2(c), the range image is split into various segments. In our prediction method, each segment is predicted using the average value of the points belonging to the cluster.

2) *Ground Prediction With RANSAC Method*: By using RANSAC [28], the points being clustered in the ground are fitted by a 3D plane. In the Cartesian coordinate system, a plane can be expressed as

$$d = ax + by + cz, \quad (4)$$

where  $(a, b, c)$  denotes the normal vector  $n$  and  $d$  represents the distance from the origin to the plane. According to the parameters of the LIDAR sensor, we calculate the virtual points' coordinates using the RANSAC-fitted plane and convert these points into a predicted range image. The difference between the real range image and the predicted range image is calculated as residual data for further processing.

3) *Prediction Result*: Through the above steps, a residual map is calculated between the range image and prediction map, which is prepared for further lossless or lossy compression. As we can see, the difference between the predicted and the actual depth map is nearly zero, as shown in Fig. 2(g). As a result, we can use few bits to encode each point, which demonstrated our proposed method can largely remove the spatial redundancies within the point cloud. In the next section, we will explore lossless and lossy coding methods for coding the contour map and residual data to reduce the coding redundancy.

#### D. Coding Method

1) *Contour Map Encoding*: In order to reconstruct the range image, we also need to encode the contour map. Therefore, an efficient contour encoding method is explored [26]. The segmentation data include the per-segment shape, and the per-region

value. The data can be better compressed by decoupling these two components.

**Boundary encoding**: With the aim of encoding the segment shape, we take the boundary pixels between the two segments into consideration. Removing the per-pixel labels, a boundary mapping is generated in which a pixel  $(x, y)$  is 1 if one of the pixels in  $(x + 1, y)$  or  $(x, y + 1)$  belongs to a different segment and 0 otherwise. The boundary map is uniformly subdivided into  $4 \times 4$  pixels macroblocks. Each block takes a random value of  $2^n$  distinct values, which needs  $n$  bits to encode without further operation.

**Per-region value encoding**: So far, we have only concentrated on transforming the boundary of the segment image. However, each region value is equally important. The boundary map splits each image into different parts. As all pixels in the same segment have the same value, only one value per segment needs to be stored.

**Exceptions**: The entire contour map can be reconstructed by the information provided in the boundary encoding part. Pixels not on a segment boundary are easily recovered according to the per-region value encoding information. However, we should pay more attention to the pixels between the two borders, which rarely take the value of their neighbors. Therefore, we construct an array individually to store the locations of these pixels and their values.

**Reconstruction**: Firstly, the boundary map is reconstructed according to the encoding data. Then, we fill the data for each segment.

2) *Residual Data Compression Method*: Several lossless encoding schemes are considered to compress the residual data after prediction. These include ZStandard, LZ4, LZ5, Lizard, Deflate, and BZip2. Besides these, we also explore traditional lossy image coding methods, like JPEG and JPEG2000, to compress the residual data.

## V. EXPERIMENTAL RESULTS

### A. Experimental Conditions and Evaluation Metrics

The proposed algorithm is implemented in C++ using some functions of the Point Cloud Library (PCL). The experiments run on an Intel Core i5-6300HQ CPU @2.30 GHz with 4-GB memory. The KITTI dataset is chosen to conduct our experiments. The raw data set consists of four scenes: City, Residential, Campus and Road. For each sequence, 100 frames are encoded.

The performance of the proposed method is evaluated in terms of compression rate and Root Mean Square Error (RMSE). The compression rate is the ratio between the compressed data size and the original one [32]. The lower the value the better the performance.

$$CR = \frac{Size_{coding}}{Size_{original}} \times 100\% \quad (5)$$

where  $CR$  represents the compression rate.  $Size_{original}$  and  $Size_{coding}$  denote the sizes of the point cloud data before and after compressed, respectively.

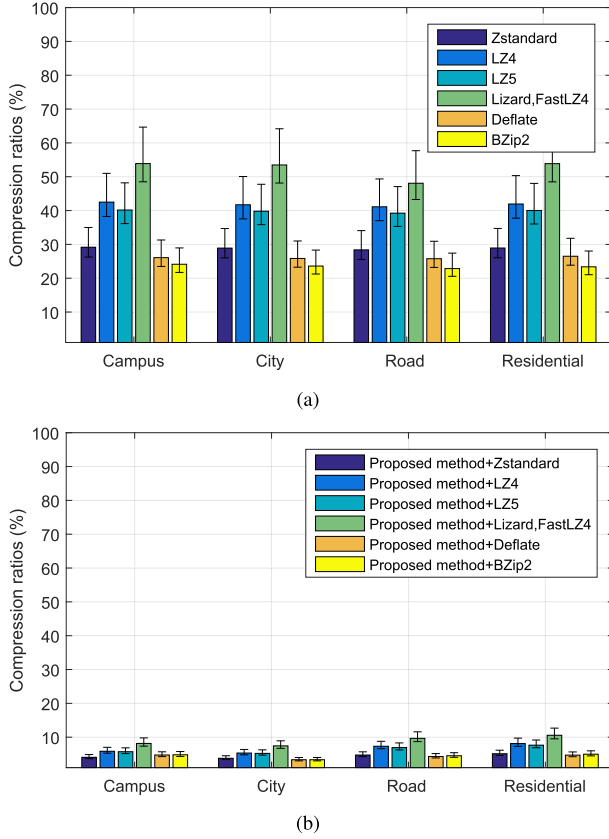


Fig. 6. Compression ratios of different lossless compression methods (a) stand-alone lossless compression (b) combination of the proposed method with lossless methods.

The RMSE represents the square root of the closest points between the original 3D point cloud and the reconstructed one, which is a measurement of reconstruction quality. Smaller RMSE values mean that the reconstructed point cloud is closer to the original point cloud. Give a  $m \times n$  range image, the RMSE is defined as follows:

$$RMSE = \sqrt{\frac{1}{m \times n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i, j) - K(i, j)]^2}, \quad (6)$$

where  $I$  and  $K$  represent the original and reconstruction range image, respectively.

### B. Lossless Compression Results

The lossless compression results are shown in Fig. 6. Fig. 6(b) depicts the average compression ratios with our proposed prediction method combined with different lossless compression schemes, including ZStandard, LZ4, LZ5, Lizard, Deflate, and BZip2. For comparison, the point clouds are also directly encoded with these lossless compression algorithms, without any processing, as shown in Fig. 6(a).

From Fig 6(a), it can be observed that the compression efficiency of stand-alone lossless methods is not as good as the one of the proposed approach. Among the lossless coding methods, the BZip2 method achieves the best performance, with a compression ratio of nearly 22%, while the performance of the

TABLE I  
COMPRESSION RATE RESULTS COMPARED WITH THE OCTREE METHOD

Scene	Lossless method (Propose method+BZip2)	Octree method		
		Distance Resolution		
		1mm <sup>3</sup>	5mm <sup>3</sup>	1cm <sup>3</sup>
Campus	7.14	20.73	12.95	10.53
City	3.31	20.14	12.70	10.28
Road	4.48	19.69	12.01	9.60
Residential	4.98	20.35	12.57	10.15

Lizard method is poor, with a compression ratio of around 50%. Lossless coding methods are designed by removing the coding redundancies of data without considering the characteristics of the point cloud structure. Therefore, the compression rates for the point clouds of different scenes hardly shows much difference.

From Fig 6 (b), as we can see, the combination of the proposed prediction approach and the lossless compression method achieves an impressive compression ratio. The smallest compression ratio is 3.3%, achieved by the combination of the proposed method with the BZip2 scheme for the point cloud of the city scene. Even the worst compression efficiency is 10.56%, achieved with the combination of the proposed method and the Lizard scheme. As the structure of point cloud data in city scenes is single, which contains a large number of ground points, the structure of contour map is simple. As a result, we can use just a few bits to encode the contour map. In contrast, the point cloud of a residential scene is complex, and the compression ratio is much higher.

Comparing Fig 6(b) with Fig 6(a), it can be seen that for the point cloud of the same scene, the combination of the proposed method with the lossless methods achieves much lower compression ratio. The above experimental results indicate that the proposed method can largely remove the spatial redundancy of the point cloud data. To achieve an optimal compression efficiency, the proposed prediction process is indispensable.

Table I depicts the compression ratio results of the proposed lossless compression method compared with the octree method [29]. The encoding precision of the octree method is set to 1 cubic millimeters, 5 cubic millimeters, and 1 cubic centimeters, respectively. It can be observed that the proposed method obtained smaller compression rate than the Octree method.

### C. Lossy Compression Results

Besides the lossless compression methods, traditional lossy image compression methods, such as JPEG and JPEG2000 [30] [31], are also explored to compress the residual data. To evaluate the performance, we compare the results of our algorithm with image-based [6] and SLAM-based [7] point cloud compression methods. The two methods are also designed for vehicle point cloud compression.

As residual data are recorded in floating-point numbers, the traditional techniques can encode only integer pixel values. In order to decrease the distance error, we try to use a recurrent data coding method which treats the error after each step as the new input and applies the JPEG or JPEG2000 coding method



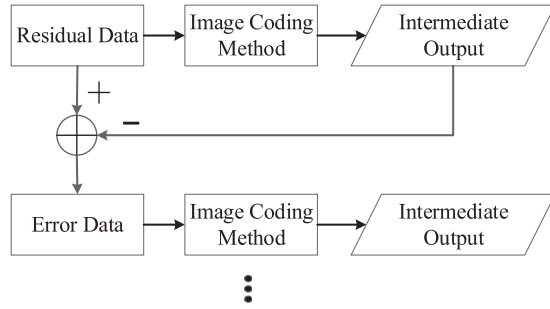
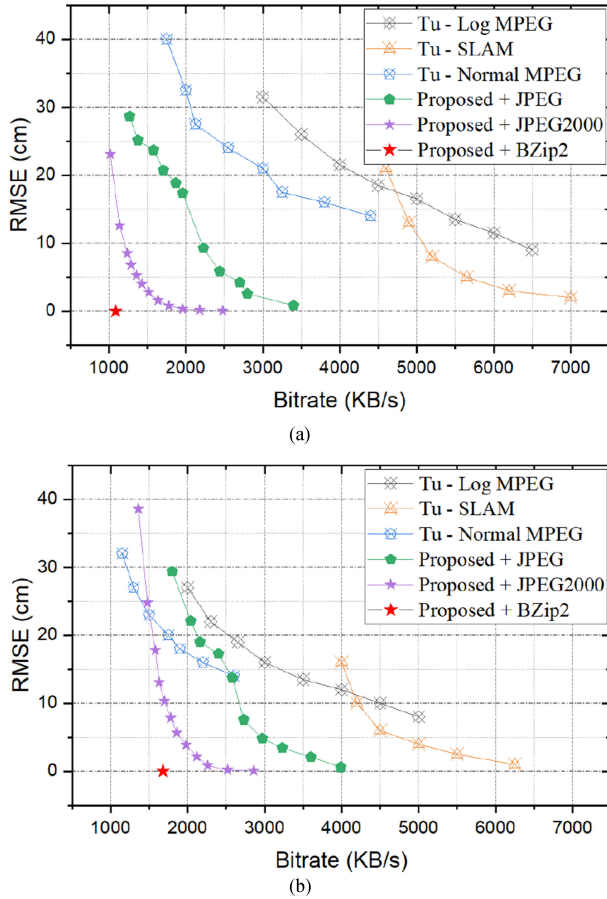


Fig. 7. Recurrent coding method.

Fig. 8. Comparison of RMSE-bitrate curves of two scenarios with Tu *et al.*'s methods: (a) city scene; (b) residential scene.

again to compress the error. This idea is illustrated in Fig. 7. This approach introduces a trade-off between distance error and the compression efficiency. In our scheme, the data are encoded twice.

Fig. 8 shows the performances of our algorithm compared with Tu *et al.*'s recently proposed methods [6], [7]. Comparison results are given in terms of the RMSE-bitrate curves of two scenarios of point cloud data. The RMSE-bitrate curves reflect the relationships between the bit rate and RMSE. A smaller bitrate and RMSE means better coding performance, as it achieves a lower RMSE with smaller bandwidth simultaneously.

It can be observed that the proposed algorithm has the outstanding advantages with bit rate and RMSE. As Tu *et al.*'s

method seldom do preprocessing for the point cloud data, there are still redundancies with the point cloud. Our proposed method needs a much lower bit rate in the small RMSE situation. In addition, because JPEG2000 works better than JPEG, the combination of the proposed method with JPEG2000 shows better performance.

In order to make an intuitionistic performance comparison with the lossless method, we also plot the best lossless compression result with a red asterisk in the figure. As the RMSE of the lossless method is zero, the curve of the lossless method is only a point. The bit rate for the city and residential scene point cloud with the lossless BZip2 method is 1090 KB/s and 1680 KB/s, respectively. As we can see, compare with the JPEG2000 method, the lossless method also shows strong competitiveness.

The above experimental results indicate that the proposed prediction method can largely remove the spatial redundancy within the point cloud, achieving a high compression efficiency by either a lossless or lossy coding method.

#### D. Contribution of Each Step for Compression Efficiency

With the aim of evaluating the contribution of each step, we record the average changes of data volumes of a frame of the point cloud during compression. BZip2 and JPEG2000 are selected as the representative methods to encode the residual data. For JPEG2000, we set the parameter CompressionRatio as 2, which means the size of the compressed data will be half of its original size.

The results are illustrated in Table II. As we can see, after the conversion, the data size is reduced to nearly a quarter of its original size. The prediction efficiency is related to the complexity of the point cloud. The data volume can be reduced to half for the point cloud with a simple structural complexity, such as city or campus scenes. The last contribution comes from the coding part. It is noteworthy that, compare with the lossless methods, the lossy JPEG2000 method hardly shows any advantage or is even worse. This is because JPEG2000 is specially designed for image compression by removing the redundancies of the color information. However, the residual data are based on distance information. Moreover, the proposed prediction step has already removed the spatial redundancy of the point cloud.

The steps of conversion, prediction, and coding are all important techniques of the proposed scheme. Their contributions are not simply added together, but they have a multiplication effect. Thus, even a small improvement in a single step can make a big difference.

#### E. Speed Performance

The proposed compression method consists of four steps, namely, range image conversion (step 1), point cloud segmentation (step 2), prediction (step 3) and coding process (step 4). We test the average speed performance of the proposed method with 100 frames point cloud sequences. The average coding times for a single point cloud of the proposed lossless and lossy methods are presented in Table III, respectively. As we can see, the total coding time is 0.502 s for the lossless method (Bzip2) and 0.378 s for the lossy method (JPEG2000).

TABLE II  
AVERAGE CHANGES IN DATA VOLUMES DURING COMPRESSION (KB)

Point Cloud Scene	Original Point Cloud Size	After Step 1 Range Image Conversion	After Step 3 Prediction	After Step 4 Coding	
				Lossless (BZip2)	Lossy (JPEG2000)
Campus	3208	835	449	150	201
City	3149	837	413	109	178
Road	3331	822	513	149	240
Residential	3222	823	537	168	212

TABLE III  
AVERAGE CODING TIME OF PROPOSED METHODS (S)

Coding method	Step 1 & Step 2	Step 3	Step 4	Total
Lossless	0.17	0.072	0.26	0.502
Lossy	0.17	0.072	0.136	0.378

## VI. CONCLUSIONS AND DISCUSSION

In this letter, an efficient compression scheme for LIDAR point cloud data, which is suitable for applications of autonomous driving, has been introduced. Experimental results demonstrate that the proposed method outperforms the other methods. The lossless method can compress the point cloud data to 1/20 of its original size without any distance distortion. It owes its performance to an efficient prediction method. To the best of our knowledge, our method is the first to extend the DMM technique in 3D-HEVC to point cloud compression. It utilizes both the advantages of the image coding method and characteristics of the point cloud. Future study will concentrate on dense 3D point cloud maps compression.

## REFERENCES

- [1] P. Yun, J. Jiao, and M. Liu, "Towards a cloud robotics platform for distributed visual SLAM," in *Proc. Int. Conf. Comput. Vis. Syst.*, 2017, pp. 3–15.
- [2] M. Liu, "Robotic online path planning on point cloud," *IEEE Trans. Cybern.*, vol. 46, no. 5, pp. 1217–1228, May 2016.
- [3] Y. Sun, M. Liu, M. Q.-H. Meng, "Motion removal for reliable RGB-D SLAM in dynamic environments," *Robot. Autom. Syst.*, vol. 108, pp. 115–128, 2018.
- [4] S. Wang, H. Huang, and M. Liu, "Simultaneous clustering classification and tracking on point clouds using Bayesian filter," in *Proc. IEEE Int. Conf. Robot. Biomimetics*, 2017, pp. 2521–2526.
- [5] J. K. Ahn, K. Lee, J. Sim, and C. 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.
- [6] C. Tu, E. Takeuchi, C. Miyajima, and K. Takeda, "Compressing continuous point cloud data using image compression methods," in *Proc. IEEE Int. Conf. Intell. Transp. Syst.*, 2016, pp. 1712–1719.
- [7] C. Tu *et al.*, "Continuous point cloud data compression using SLAM based prediction," in *Proc. Intell. Veh. Symp.*, 2017, pp. 1744–1751.
- [8] 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.*, 2015, pp. 1–8.
- [9] X. Wang, Y. A. Şekercioglu, T. Drummond, E. Natalizio, I. Fantoni, and V. Frémont, "Fast depth video compression for mobile RGB-D sensors," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 4, pp. 673–686, Apr. 2016.
- [10] C. Yang *et al.*, "Fast depth map coding based on virtual view quality," *Image Commun.*, vol. 47, no. C, pp. 183–192, 2016.
- [11] V. Morell *et al.*, "Geometric 3D point cloud compression," *Pattern Recognit. Lett.*, vol. 50, no. C, pp. 55–62, 2014.
- [12] J. Elseberg, D. Borrmann, and A. Nuchter, "One billion points in the cloud—an octree for efficient processing of 3D laser scans," *ISPRS J. Photogrammetry Remote Sens.*, vol. 76, no. 1, pp. 76–88, 2013.
- [13] 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.*, 2012, pp. 778–785.
- [14] X. Zhang, W. Wan, and X. An, *Clustering and DCT Based Color Point Cloud Compression*. Norwell, MA, USA: Kluwer, 2017.
- [15] T. Golla and R. Klein, "Real-time point compression," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 5087–5092.
- [16] S. Gumhold, Z. Kami, M. Isenburg, and H. Seidel, "Predictive point-cloud compression," in *Proc. ACM SIGGRAPH Sketches*, Los Angeles, CA, USA, 2005, Paper no. 137.
- [17] R. Schnabel and R. Klein, "Octree-based point-cloud compression," in *Proc. 3rd Eurographics Symp. Point-Based Graph.*, Zurich, Switzerland, 2006, pp. 111–120.
- [18] 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.
- [19] C. Zhang, D. Florncio, and C. Loop, "Point cloud attribute compression with graph transform," in *Proc. IEEE Int. Conf. Image Process.*, 2014, pp. 2066–2070.
- [20] 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.*, 2014, pp. 6152–6156.
- [21] R. Mekuria, K. Blom, and P. Cesar, "Design, implementation and evaluation of a point cloud codec for tele-immersive video," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 4, pp. 828–842, Apr. 2017.
- [22] A. Geiger *et al.*, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013.
- [23] I. Bogoslavskyi and C. Stachniss, "Fast range image-based segmentation of sparse 3D laser scans for online operation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 163–169.
- [24] I. Bogoslavskyi and C. Stachniss, "Efficient online segmentation for sparse 3D laser scans," *J. Photogrammetry Remote Sens. Geoinformation Sci.*, vol. 85, no. 1, pp. 41–52, 2017.
- [25] K. Miller *et al.*, "3D high-efficiency video coding for multi-view video and depth data," *IEEE Trans. Image Process.*, vol. 22, no. 9, pp. 3366–3378, Sep. 2013.
- [26] B. Matejek *et al.*, "Compresso: Efficient compression of segmentation data for connectomics," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assisted Intervention*, 2017, pp. 781–788.
- [27] M. J. Knieser *et al.*, "A technique for high ratio LZW compression," in *Proc. Conf. Des., Autom. Test Europe Exhib.*, 2003, pp. 116–121.
- [28] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.
- [29] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 1–4.
- [30] G. K. Wallace, "The JPEG still picture compression standard," *Commun. ACM*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.
- [31] B. Martin, C. Christopoulos, and E. Majani, "Information technology: JPEG 2000 image coding system," *ISO/IEC*, 2000, Art. no. 15444-1.
- [32] L. Wang *et al.*, "Point-Cloud compression using data independent method-A 3D discrete cosine transform approach," in *Proc. IEEE Int. Conf. Inf. Autom.*, 2017, pp. 1–6.