2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)
Windsor Oceanico Hotel, Rio de Janeiro, Brazil, November 1-4, 2016

Compressing Continuous Point Cloud Data Using Image Compression Methods*

Chenxi Tu[1] Eijiro Takeuchi[2] Chiyomi Miyajima[3] Kazuya Takeda[4]

Abstract— **Continuous point cloud data has become very important recently as a key component in the development of autonomous driving technology, and has in fact become indispensable for some autonomous driving applications such as obstacle detection. However, such large amounts of data are very expensive to store and are difficult to share directly due to their volume. Previous studies have explored various methods of compressing point cloud data directly by converting it into 2D images or by using tree-based approaches. In this study, rather than compress point cloud data directly, we choose to compress packet data which is raw point cloud data, by converting it losslessly into range images, and then using various image/video compression algorithms to reduce the volume of the data. In this paper, four methods, which are based on MPEG, JPEG and two kinds of preprocessing approaches for each method, are evaluated for range images compression. PSNR V.S Bitrate and RMSE V.S Bitrate are used to compare and evaluate the performance of these four methods. As whether a lossy compression methods is good always depending on the application, a localization application to test point cloud data reconstruction performance is also conducted. By comparing these methods, some important conclusions can be got.**

I. INTRODUCTION

Autonomous driving technology is developing rapidly. Four years ago, Google was granted the first self-driving car license in Nevada, and as the technology becomes more and more mature, the company plans to begin testing over 100 autonomous drive vehicles in several cities. Previous obstacles caused by hardware issues should be resolved in the near future, as sensors such as LiDAR (Light Detection and Ranging), mounted on top of vehicles, become more widely available. LiDAR systems for autonomous driving used to be very expensive (around $100,000 USD), however more and more companies are now producing cheaper models. The Velodyne Company's VLP-16 is priced around $8,000 USD, and Quanergy's M8-1 may even be available in the $1,000 USD range when mass produced, allowing autonomous driving to become a widely used technology in the near future.

Autonomous driving technology and other automated assistance systems will need to process huge amounts of point cloud data [1][2]. A Velodyne HDL-64S2 LiDAR system, when set to revolve at 10Hz, stores around 3 MB of data per frame every 0.1 seconds. Under such conditions, one hour of point cloud driving data should be over 100GB. LiDAR data is most often used for real-time decision making during autonomous driving, but sharing or storing this data also allows many other useful applications, such as interactive traffic monitoring, or accident investigation. Using current data storage and transmission technology, this is without a doubt too much data to store or transmit, thus compressing point cloud data is an absolutely necessary task. Point cloud data compression can be divided into two types of tasks. One task is the compression of a static point cloud, which is widely used in 3D mapping, and the other task is compression of streaming point clouds. This study will focus on the latter.

There are two general choices when performing a compression task; lossy or lossless methods. This study will explore both methods. The optimum compression method always depends on the specific application. For example, JPEG and MPEG compression methods are most often used for applications which involve human sensory perception, and thus they are designed to perform well enough to "fool" the human senses rather than to perfectly reproduce the original data.

This paper evaluates point cloud compression performance using several image compression methods to validate these capability for point cloud compression whose reconstructed data aim to be used in autonomous driving application. The evaluated compression methods are MPEG, JPEG and two kinds of preprocessing method for each method. Finally, the paper illustrates the evaluated results of quality of reconstructed data and localization quality that used in autonomous driving system.

II. RELATED WORK

A number of studies on compressing point cloud data have been published. They can be divided into two groups by target (static point clouds and streaming point clouds), or by algorithm (height/occupancy map and tree based approaches).

Static point cloud compression is a much more popular research topic and a number of methods have been proposed using height map encoding over a planar domain. Pauly [7] may have been the first to introduce height map encoding to a point cloud by doing a spectral analysis of a point cloud. Several approaches based on this method have been published. Hubo [8] performed a compression task using

*This work was not supported by any organization

[1] Chenxi Tu is with Media Science, Informarion Science School, Nagoya University, Furo-cho, Chikusa-ku, Nagoya 464-8603 Japan tu.chenxi@g.sp.m.is.nagoya-u.ac.jp

[2] Eijiro Takeuchi is with Media Science, Informarion Science School, Nagoya University, Furo-cho, Chikusa-ku, Nagoya 464-8603 Japan takeuchi@coi.nagoya-u.ac.jp

[3] Chiyomi Miyajima is with Media Science, Informarion Science School, Nagoya University, Furo-cho, Chikusa-ku, Nagoya 464-8603 Japan miyajima@nagoya-u.jp

[4] Kazuya Takeda is with Media Science, Informarion Science School, Nagoya University, Furo-cho, Chikusa-ku, Nagoya 464-8603 Japan kazuya.takeda@nagoya-u.jp

representative height maps to represent all of the targeted height maps. Ochotta [9] compressed point cloud data using a wavelet transform on height maps and binary masks (which were, in fact, occupancy maps) which created patch shapes which were then compressed using arithmetic coding. An improved version of this method [10] employs a generalized Lloyd algorithm [11] to obtain fewer patches and achieve a lower error rate. Schnabel [12][13] used height maps of geometric primitives, which are non-planar domains such as planes, spheres, cylinders, cones and tori, and used vector quantization to compress height maps.

Golla and Klein [14] developed a real-time method which split data into compression chunks, decomposing a world space into voxels of a user-specified size, then, using a data-adaptive approach based on an octree, chunks are divided into patches. Height and occupancy maps are generated in each patch and compressed using JPEG, achieving a state-of-the-art compression ratio.

Some tree-based methods have also been proposed. Schnabel and Elseberg [15][16] use an Octree-based method specifically designed for point sampled geometry and are based on local surface approximations. While S. Gumhold [17] use a stream predictive tress. One big advantage of tree-based methods is that octree-based methods can be lossless. In addition, Hubo [18] use a kd-tree and allow random access.

Some of these proposed methods can be applied to streaming point clouds, such as [17]. Kammerl[19] is real-time method which can get a good compression radio and coordinate precision for streams of point cloud data with a fixed-position camera by reducing the redundancy between neighboring frame.

In contrast, in this study we focus on point cloud data that obtained at driving scene. Previous studies focus more attention on static point clouds which record one object, such as a building, a human face, a work of art, etc. In contrast, continuous point clouds that obtained by autonomous vehicles produce streams which are sparser and have longer ranges (from 0 cm to 13,000 cm, for example). In this study, rather than compressing point cloud data in its final format, our goal is to compress the original, raw data of point cloud - i.e., the packet data (also known as velodyne raw data). In order to avoid confusion, we will refer this data as "packet data" in the rest of this paper. Packet data can be used to obtain range images losslessly with a mapping rule. As a result, the data can be changed into a image stream, which can then be compressed further by using image compression algorithms.

III. FRAMEWORK AND DATA CONVERSION

The overview of the target system is shown in Fig.1. Fig. 2 shows details of compression and decompression part of Fig.1. As shown in Fig.2, "Compress" can be divided into two parts: Compressing rotation position vectors which represent the yaw angle of each point; Compressing range images which contain the distance and pitch angle information of each point. This section describes details of

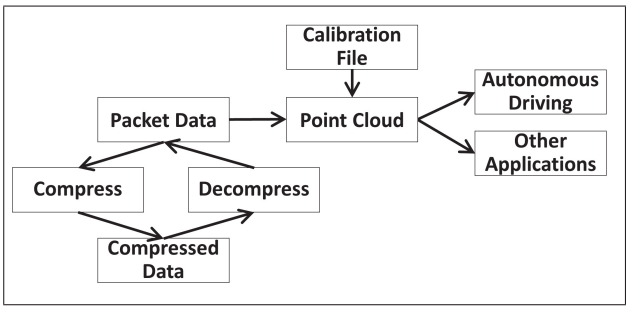

Fig. 1. Overview of whole system. This research focus on compress and decompress part. In localization error evaluation part, we will reconstruct point cloud data and use a localization application to test its performance.

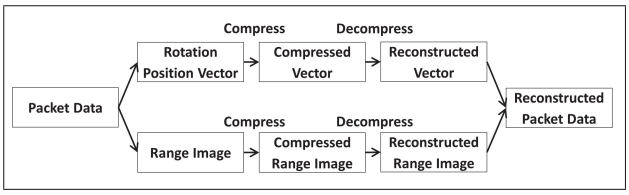

Fig. 2. Compressing process can be divided into two parts: rotation position compression and range image compression

target packet data format, data representation, range image conversion, and compression method for rotation position vector.

A. Data Format of Packet Data

In this section, we use Velodyne HDL-64S2 LiDAR as an example to introduce the packet data's format and how to get range image and rotation position vetor. In order to use image/video compression methods, packet data should first be converted into range images with rotation position vector. Fortunately, the structure of packet data is already naturally the same as range images. In LiDAR packet data, each point is represented by three types of information: ID, rotation position and distance.

- ID represents a LiDAR laser ID. The LiDAR used here is Velodyne HDL-64S2, which has 64 lasers, so ID is a number from 0 to 63. Because each laser is fixed at a specific angle, ID can also be used to represent the pitch angle.
- Rotation position refers to the position of the LiDAR apparatus as the device is revolving. Here LiDAR use 0 to 35,999 to represent a point on a 360° circle, thus it can be seen as directly representing yaw angle.
- Distance is the distance from point to LiDAR. The range of it is from 0cm to 130000cm

When dealing with streaming point cloud, the unit is always one frame, which means one 360° rotation of the LiDAR scanner (which is in fact a little more than 360°). Thus, when dealing with packet data, the unit is one scan, which corresponds to one frame in the point cloud. The data format is shown in Fig 3. When using the HDL-64 LiDAR, one scan contains 384 packets, each packet contains 12 blocks, and each block has 32 points per emission (equal to half of

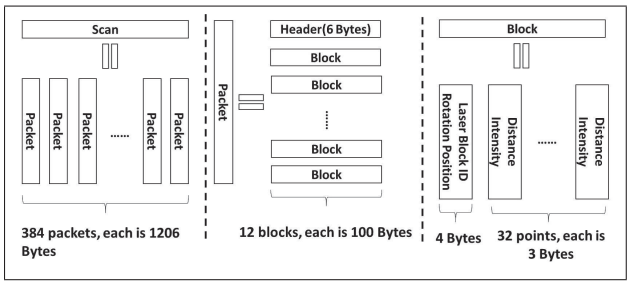

Fig. 3. Date format of packet data from Velodyne HDL-64S2 LiDAR sensor

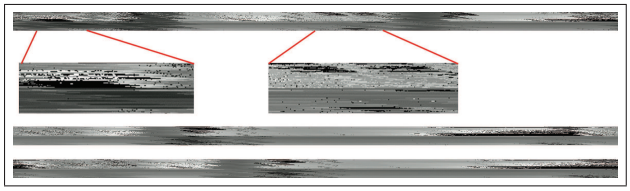

Fig. 4. Examples of range images

one emission, because of the 64 laser receivers are divided into two parts: Upper parts and Lower part), so one scan contains 133,632 points and includes $384 \times 12/2 = 2088$ times emissions

B. Range Image Conversion

Data from each scan can be converted into one 64×2088 range image. ID information can be naturally represented by the row number of the range image, each column corresponds to one scan emission, and pixel information represents distance information. Table 1 shows the mapping conversion rules and Fig. 4 shows examples of range images. By doing this, rotation position attached to each emitting still need to be stored. It should be noted that for different scans, the rotation position attached to each emission are also different and according to rotation position, one scan do not represent $360°$ exactly (more than $360°$, in fact.) Because of this, in order to compress and reconstruct packet data, not only the range image but also the 2,088 rotation positions of each scan should be stored and compressed.

TABLE I
CONVERTING RULE

Distance of (0,1)	Distance of (0,2)	...	Distance of (0,2088)
Distance of (1,1)	Distance of (1,2)	...	Distance of (1,2088)
...	...	...	...
Distance of (63,1)	Distance of (63,2)	...	Distance of (63,2088)

C. Storing and Compressing Rotation Position

Here, go on using Velodyne HDL-64S2 as example. In this case, one scan contain 2088 times emitting, but the interval of rotation is not fixed. In most case, the interval is $0.18°$ (which mesns after one emission, LiDAR rotates $0.18°$ and then emits again) .In a few cases, the interval is $0.09°$, and the position is not fixed. Taking advantage of this property, we

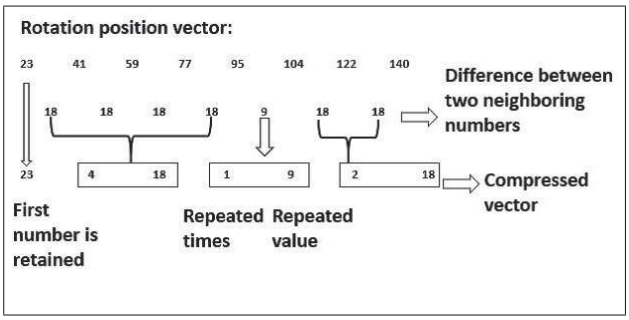

Fig. 5. Run Length Encoding. At first, calculate the difference of adjacent elements of vector. Then use the repeating times and repeating values to represent repeating numbers. In rotation position vector, the most differences of adjacent elements are same which lead to a good compression result of Run Length Encoding method.

can use Run Length Encoding, which is a lossless method, to compress each row. Run Length Encoding uses repeating values and times to represent repeating numbers. Details are shown in Fig. 5. Take a n seconds driving point cloud data, which contains $10n$ scans (when the LiDAR's rotating speed is set at 10 Hz), as an example. Each scan has a rotation vector with 2088 elements, so the whole data results in a $10n \times 2088$ matrix. By using Run Length Encoding, we can compress rotation matrix file from $0.29n$ Mb to around $0.01n$ Mb which is small enough.

IV. RANGE IMAGE COMPRESSION

Compressing streaming range image is the main and key part of our approach. There are two key components of streaming data compression: quantization and prediction. Quantization, which is the core of most compression methods, is the process of describing a large set of input values with a smaller subset. Prediction refers to the use of previous data or future data,even the differences between current and previous or future data, to estimate current data. Prediction is widely used in many stream data compression methods.

In this section, four different methods will are tested: MPEG with range image sequence, MPEG with log range image sequence, JPEG with range image sequence, JPEG with layered range image sequence. MPEG [3][4] is a typical data compression method which utilizes quantization and prediction components. It is one of the most popular video (image stream) compression methods, allowing a high compression rate by using a linear prediction method. In this paper, the MPEG-4 method is used as one approach. In addition, we also use MPEG-4 on the log of range images, which involves taking the logarithm of each pixel of a range image. This is a popular audio coding strategy (like ADPCM) which can achieve high accuracy in the "high density" region. We also use compression methods based on JPEG [5][6] in this study, which allow a high compression rate for images by reducing their high frequency information. One method is to use JPEG directly on range images. Another is to rebuild range images so that rows correspond to a time sequence, and then using JPEG to compress those images.

Comparing using MPEG and JPEG, we can check the

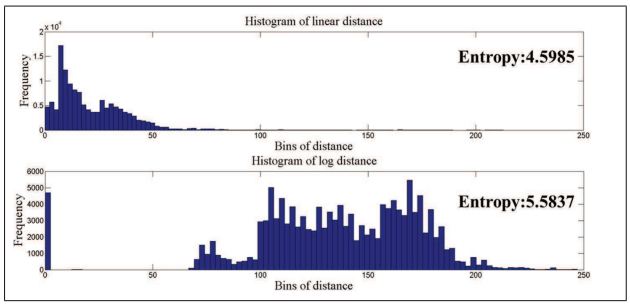

Fig. 6. Comparison of histograms of normal distance and log distance. All distance values are converted into same bits. The upper section shows normal distance and the lower section shows log distance.By using log, the distance value can distribute in a bigger range.

advantage of prediction; Comparing MPEG with range image and MPEG with log range image, we can check the effect of utilizing the distribution of range information to help compressing by using log value; Comparing JPEG with layered range image and JPEG with range image, we can check whether there is a better range data representation method for compression method then we can get better performance.

A. MPEG-4 with Range Image Sequence

After obtaining a range image sequence, MPEG-4 can be used directly to compress the data. The MPEG-4 standard is a popular, state-of-the-art video compression method. Since we are now working with range images, the stream point cloud has already become a kind of video, so MPEG-4 compression is a natural choice. MPEG-4 divides frames into three categories: I frames, P frames and B frames. I frames represent Intra-coded pictures, which are compressed by reducing spatial redundancy. P frames (Predictive-coded pictures) use previous frames to predict the current frame, while B frames (Bi-directionally predicted pictures) use both previous and future frames to predict the current frame. By using previous and future frames, temporal redundancy can be reduced during prediction.

In rest part of this paper, we call this method "normal MPEG" for short.

B. MPEG-4 with Log Range Image Sequence

In reality, although LiDAR can detect such a long range, the most points gather at the nearby areas (less than 30 meters away from vehicle) especially driving in the city, as what the upper picture in Fig 6, shows. So by using the log value, higher accuracy can be got in the nearby area, which is a popular strategy in audio coding. Therefore, we also test MPEG on the log of the distance. The new log distance $d^{'}$:

$$d^{'} = |\log_2(d+1) - 5| \tag{1}$$

The reason minus 5 is included in this expression is because at very short distances there should be no data.

In rest part of this paper, we call this method "log MEPG" for short.

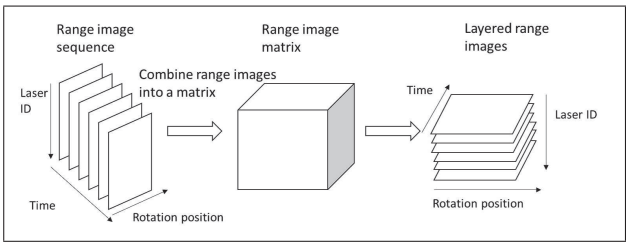

Fig. 7. Re-cutting the range image matrix

C. JPEG with Range Image Sequence

Another compression method which was tested is one base on JPEG, which focuses on reducing spatial redundancy. By using a discrete cosine transform (DCT) or wavelet transform, images can be converted into the frequency-domain. Compression can then be performed by reducing high frequency information. Thus, one obvious compression method is to use JPEG to directly compress the range images.

In rest part of this paper, we call this method "normal JPEG" for short.

D. JPEG with Layered Range Image

Rather than using JPEG directly we can also combine these sequences together to form a matrix and then re-cut the matrix by rows (layers), i.e., we can scan each laser emission by time sequence. To a n seconds driving point cloud data acquired by Velodyne HDL-64S2, as a result, 64 layer range images, corresponding to the 64 laser sensors of the LiDAR system, can be obtained. The size of each image is $2088 \times 10n$ and the conventional JPEG method can then be used for compression. The reason why we do not use JPEG to directly compress the range images is that JPEG compresses images by ignoring high frequency information. High frequency information in images represents details of the image such as boundaries or strong grayscale contrast. Range images have large amounts of high frequency content, as they come from packet data which has not been processed with a calibration file. So directly using JPEG to compress range images will result in the loss large amounts of information. On the other hand, layered range images contain less high frequency information. So by using layered range image, more information can be kept which is the purpose of this method. Temporal redundancy can also be reduced by using this method since the vertical axis of layered range images represents a time series.

In rest part of this paper, we call this method "layer JPEG" for short.

V. EVALUATION

In order to evaluate the performance of these four approaches, we performed two kinds of evaluations. The first evaluation uses Peak Signal to Noise Ratio (PSNR) and Root-Mean-Square-Error (RMSE), both of which are very important indicators of image and point cloud compression, as well as V.S. Bitrate, to show how well each approach works. In addition, because the quality of a compression

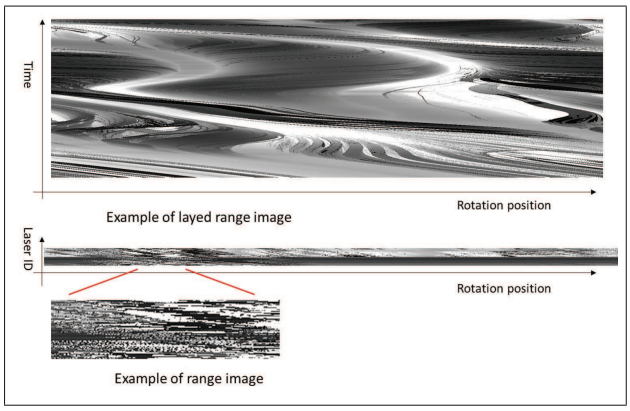

Fig. 8. Contrast between a layered range image and a range image. No doubt that range image is a little "chaotic" with more high frequency information while layed range image is smoother with less high frequency information because its row means time sequence.

task always depends on the application that the compressed data will be used for, we perform a second evaluation based on a localization application. This will show how well reconstructed (i.e., decompressed) point clouds can perform after being compressed using each method. This evaluation involves three samples of driving point cloud data, each of which is one minute in length, recorded in three different situations: driving in a parking lot; driving on an urban road (wide roads, traffic lights, intersections, a tower, etc.); driving in a typical Japanese residential/uptown area (narrow roads, most buildings are two or three floors, etc.).

A. Range Error Evaluation

In this evaluation, we reconsturct packet data of compressed data and compare the performance of our four compression methods:

- Normal MPEG: Directly compresses the range images using MPEG.
- Log MPEG: Changes the values of pixels into their logarithmic values, and then uses MPEG for compression.
- Normal JPEG: Directly compresses the range images using JPEG.
- Layer JPEG: Re-scans the range images layer by layer, and then uses JPEG to compress the layered range images.

PSNR V.S. Bitrate and RMSE V.S. Bitrate are used to evaluate compression performance. Given a $m \times n$ image I and its noisy approximation K

$$\text{MSE} = \frac{1}{mn} \sum_{m-1}^{i=0} \sum_{n-1}^{j=0} [I(i,j) - K(i,j)]^2 \quad (1)$$

$$\text{RMSE} = \sqrt{\text{MSE}} \quad (2)$$

$$\text{PSNR} = 20 \times \log_{10}(\frac{\text{MAX}_\text{I}}{\text{RMSE}}) \quad (3)$$

$\text{MAX}_\text{I}$is the maximum possible pixel value of the image. Here the maximum of pixel means the maximum distance can be detected by sensor which is 13000 cm. So here MAX_I is 13000.

Fig. 9 shows the results of our first evaluation in the three different scenarios. Note that performance varies in each scenario. For example, for the same PSNR and RMSE, the parking lot scenario required a little higher bitrate than the residential area (espically MPEG based methods).

However, in general the results are almost the same in all three situations. At lower bitrates, normal MPEG achieves the best performance. But log MPEG do not show a better performance than normal MPEG at the same bitrate, perhaps because at distant area there are still some data points. By using logarithmic values, errors in closer areas are reduced while errors in distant areas are amplified.

At higher bitrates, layer JPEG achieves the highest performance. Layer JPEG performs better than normal JPEG in lossy compression, which means our strategy working. While under lossless conditions, normal JPEG outperformed Layer JPEG. Both methods are much more efficient than lossless compression of binary files using conventional zipping programs.

In addition, because MPEG work better than JPEG at small bitrate, we can say that for high compression rate, prediction process is indispensable

B. Localization Error Evaluation

Since compression performance is always related to the specific task it is used with, we conducted a localization to test the performance of each method when streaming cloud point data is decompressed.

Fig.10 shows the reconstructed point cloud data intuitionally. When compared with the original data, many gaps can be seen in the data compressed using the normal MPEG method. However, even under the low PSNR condition, the outline of the obstacle can still somehow be recognized by the human eye. In the case of the log MPEG method, it is clear that the area close to origin of the coordinates is more similar to original point cloud than when using other methods, which supports our theoretical hypothesis. The recovery results using JPEG seem harder to recognize in the low PSNR condition than when using MPEG. The main reason may be that the JPEG compression method gives up high frequency information, leading to indistinct details.

Next, we used a localization application to test how well the reconstructed point clouds functioned. Fig. 11 shows mapping results using original point cloud data and localization results. The mapping and localization is realized Normal Distributions Transform (NDT) scan-matching method [20][21]. The method is used in open-source autonomous driving system [22]. The localization results are calculated by NDT scan-matching method,3D map that created by original data, and each compressed data. When compared with Fig. 9, which shows the RMSE of our reconstructed data, some differences can be observed.

First, generally speaking, RMSE for the localization application is much smaller than RMSE for the reconstructed data. In other words, even reconstructed data which have large RMSE can produce very good localization results.

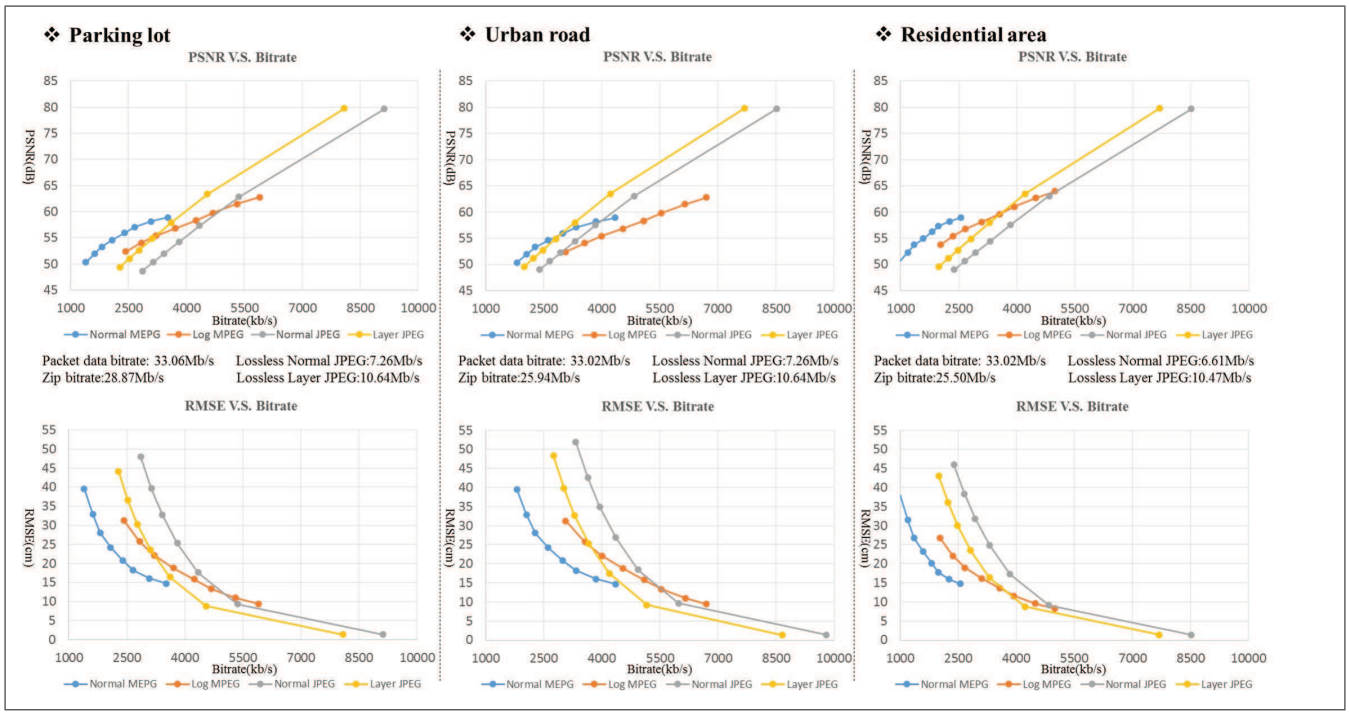

Fig. 9. PSNR/RMSE V.S. Bitrate in three different scenarios

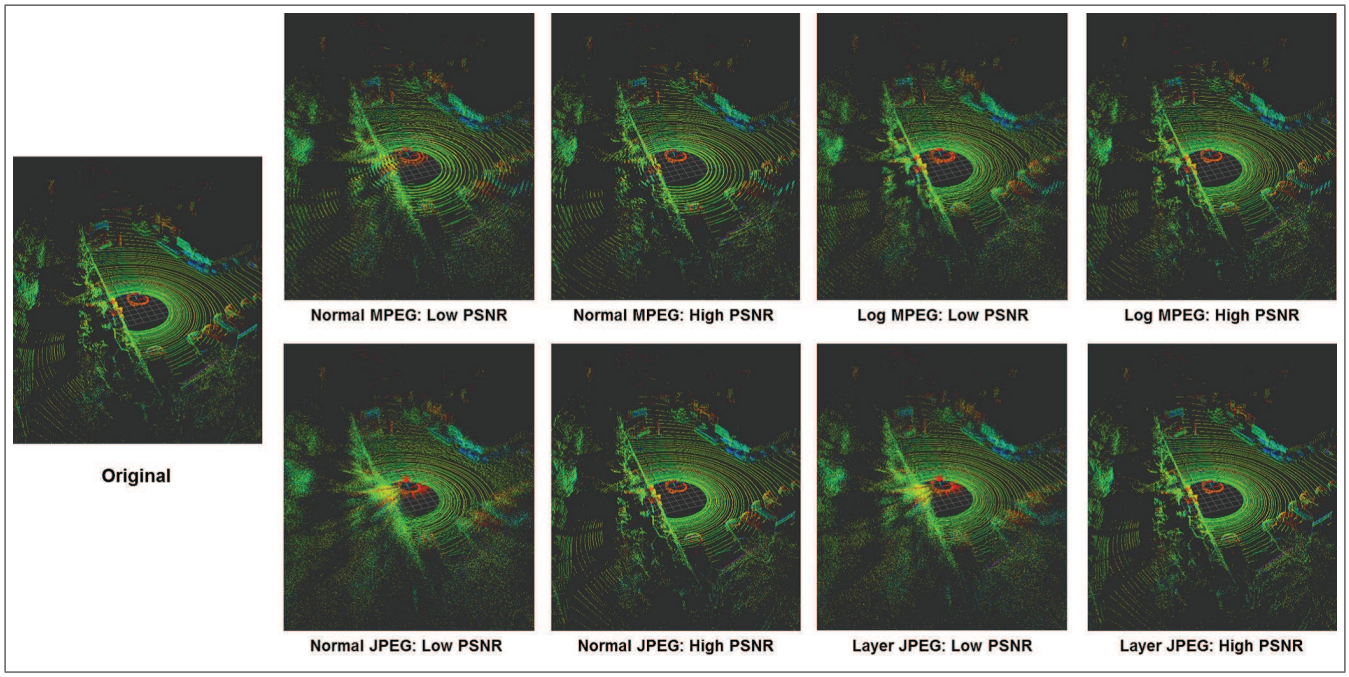

Fig. 10. Reconstructed point cloud data from different methods in the urban situation. Low PSNR and high PSNR are corresponding to lowest and highest PSNR in Fig.9

Secondly, in Fig. 11 we can see that RMSE again varies widely between the three different driving situations. RMSE in the urban road scenario is the much smaller than in the residential area and parking lot scenarios. These kinds of differences were much bigger in Fig. 9

Thirdly, when comparing our different compression methods within the same driving situations, we can see that the relative performances of normal MEPG, normal JPEG and layer JPEG in Fig. 11 remain almost the same as in Fig. 9. Only log MPEG performed worse in the localization application. In fact, on the urban road, log MPEG even became the worst performing method. The likely reason is that, while using log values can reduce point errors in nearby areas, at the same time this method amplifies point errors

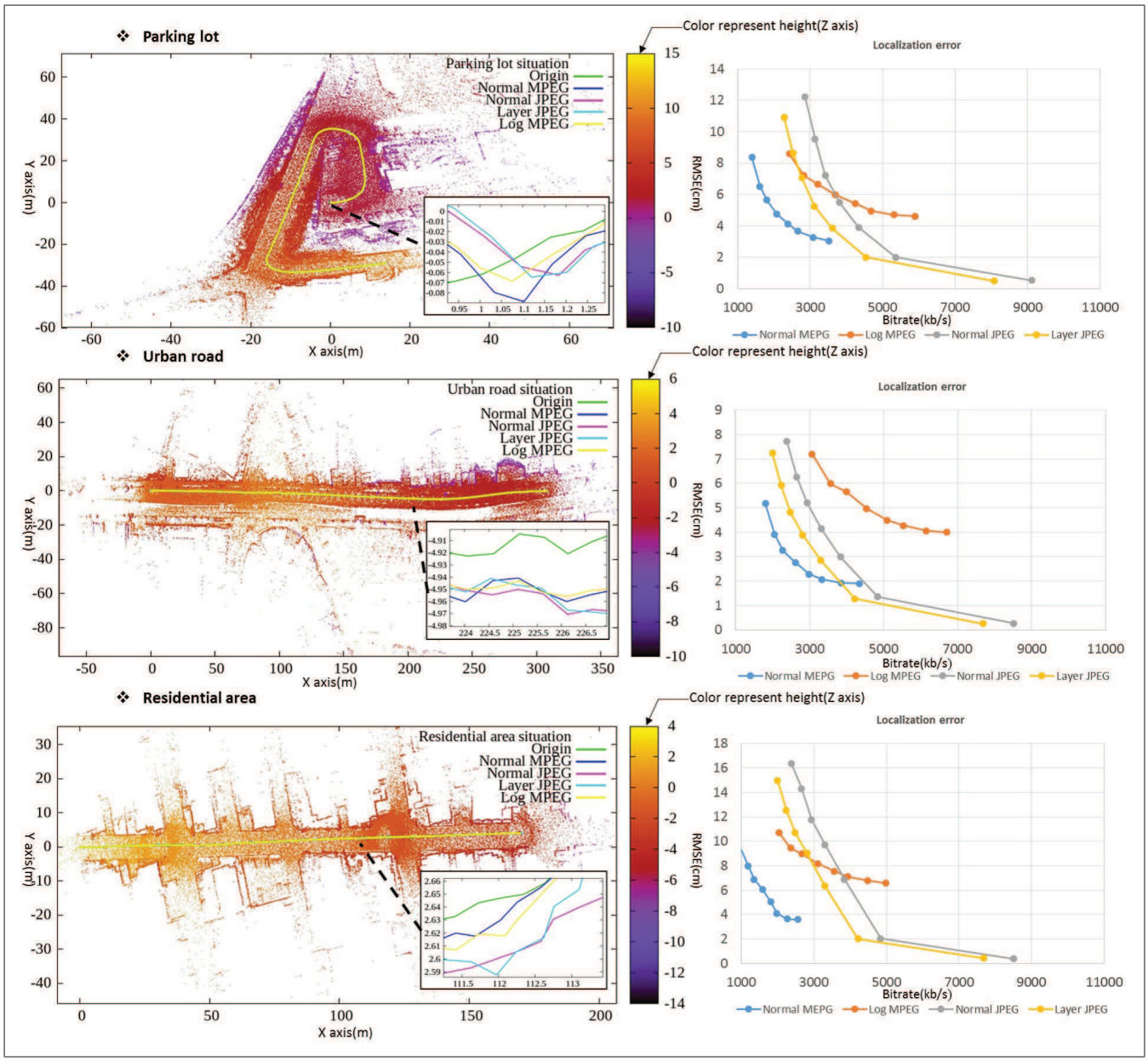

Fig. 11. The left part shows the 3D map with the localization result of original data and all four methods' reconstructed data at low PSNR. A expanded small region located by black line of dashes is also attached in three situations. Here setting localization result of original data as ground truth, the error is calculated by distance from original trajectory. The right part shows the RMSE V.S. Bitrate of localization application in three different situations.

in distant areas. Since the localization method used here depends more on the points in distant areas, this leads to poor performance of log MPEG in the localization task.

VI. CONCLUSION

This paper evaluates point cloud compression performance using several image compression methods to validate these capability for point cloud compression whose reconstructed data aim to be used in autonomous driving application. The paper proposes to separate compression processes to rotation angle data and range image data. The run length encoding is used for rotation angle compression. The MPEG, JPEG and two kinds of preprocessing approaches for each method are evaluated for range images compression.

The evaluation results of quality of reconstructed data is that at low PSNR situation, MPEG works best. This fact indicates the importance of prediction in our task. Log MPEG do not work as well as we expect, although it can reduce the error in nearby area. At low PSNR situation, layer JPEG works best, which means our re-cutting approach working. However if asking for lossless compression, normal JPEG perform better. In addition, in the different situations, the bitrates need for the same method are different.

The evaluation results of localization quality is similar to quality of reconstructed data expect for two difference. At first using the same method, the RMSE of localization is much smaller than RMSE of reconstructed data,

which means localization application can bear low quality reconstructed data. Secondly, log MPEG performed badly, considering its RMSE of reconstructed data. The reason is that using log amplifies error for distant points, and the localization application used here depends more on distant points. This fact once again confirming the theory that the performance of lossy compression methods always depends on specific applications

Generally speaking, while ask for small bitrate, the MPEG compression quality is best in tests. If we arrow 5cm localization error in RMSE, then the bit rate is 1.5Mbps, which is much smaller than nowadays average upload speed of 4G LTE, with MPEG compression and 660MB for 1 hour data. That is almost $1/150$ of point cloud data.

This paper illustrates the image compression methods are capable to be used on point cloud compression. However, the image compression methods are designed for image and human recognition. In future work, we will design more effective prediction and quantize methods for autonomous driving systems.

REFERENCES

[1] Levinson J, Askeland J, Becker J, et al. Towards fully autonomous driving: Systems and algorithms. *Intelligent Vehicles Symposium (IV)*, 2011 IEEE. pp. 163-168.

[2] Glaser C, Michalke T P, Burkle L, et al. Environment perception for inner-city driver assistance and highly-automated driving. *Intelligent Vehicles Symposium Proceedings*, 2014 IEEE. pp.1270-1275.

[3] Le Gall D. MPEG: A video compression standard for multimedia applications. *Communications of the ACM*, 1991, 34(4): 46-58.

[4] Marpe D, Wiegand T, Sullivan G J. The H. 264/MPEG4 advanced video coding standard and its applications. *Communications Magazine*, IEEE, 2006, 44(8): 134-143.

[5] Wallace G K. The JPEG still picture compression standard. *Consumer Electronics, IEEE Transactions on*, 1992, 38(1): xviii-xxxiv.

[6] Marcellin M W, Gormish M J, Bilgin A, et al. An overview of JPEG-2000. *Data Compression Conference, 2000. Proceedings. DCC 2000.* IEEE, 2000: 523-541.

[7] Pauly M, Gross M. Spectral processing of point-sampled geometry. *Proceedings of the 28th annual conference on Computer graphics and interactive techniques.* ACM, 2001: 379-386.

[8] Hubo E, Mertens T, Haber T, et al. Self-similarity based compression of point set surfaces with application to ray tracing. *Computers & Graphics,* 2008, 32(2): 221-234.

[9] Ochotta T, Saupe D. *Compression of point-based 3D models by shape-adaptive wavelet coding of multi-height fields*. 2004.

[10] Ochotta T, Saupe D. Image Based Surface Compression. *Computer graphics forum. Blackwell Publishing Ltd,* 2008, 27(6): 1647-1663.

[11] Lloyd S P. Least squares quantization in PCM. *Information Theory, IEEE Transactions on,* 1982, 28(2): 129-137.

[12] Schnabel R, Möser S, Klein R. A Parallelly Decodeable Compression Scheme for Efficient Point-Cloud Rendering. *SPBG.* 2007: 119-128.

[13] Schnabel R, Möser S, Klein R. Fast vector quantization for efficient rendering of compressed point-clouds. *Computers & Graphics*, 2008, 32(2): 246-259.

[14] Golla T, Klein R. Real-time point cloud compression. *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015: 5087-5092.

[15] Schnabel R, Klein R. Octree-based Point-Cloud Compression. *SPBG*. 2006: 111-120.

[16] Elseberg J, Borrmann D, Nchter A. One billion points in the cloud-an octree for efficient processing of 3D laser scans. *ISPRS Journal of Photogrammetry and Remote Sensing*, 2013, 76: 76-88.

[17] Gumhold S, Kami Z, Isenburg M, et al. Predictive point-cloud compression. *ACM SIGGRAPH 2005 Sketches*. ACM, 2005: 137.

[18] Hubo E, Mertens T, Haber T, et al. The quantized kd-tree: Efficient ray tracing of compressed point clouds. *Interactive Ray Tracing 2006, IEEE Symposium on*. IEEE, 2006: 105-113.

[19] Kammerl J, Blodow N, Rusu R B, et al. Real-time compression of point cloud streams. *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012: 778-785.

[20] Eijiro Takeuchi, and Takashi Tsubouchi. A 3-D scan matching using improved 3-D normal distributions transform for mobile robotic mapping. *Proceedings of the IEEE International Conference on Intelligent Robots and Systems(IROS)*, pp. 3068 -3073, 2006.

[21] Eijiro Takeuchi, Yoshiki Ninomiya and Shinpei Kato. Lane Visibility Check Methods based on High Precision Maps and 3D LiDAR for Traffic Prediction. *FAST-zero 2015 symposium*, 2015

[22] Shinpei Kato, Eijiro Takeuchi, Yoshio Ishiguro, Yoshiki Ninomiyay, Kazuya Takeda,and Tsuyoshi Hamada. An Open Approach to Autonomous Vehicles. *2015 Nov/Dec issue of IEEE Micro with a special feature on Cool Chips*, 2015