https://doi.org/10.2352/ISSN.2470-1173.2019.15.AVM-043

Image-based Compression of Lidar Sensor Data

Peter van Beek; Intel & Mobileye Silicon Valley Innovation Center; San Jose, California, USA

Abstract

Our goal is to develop methods for lossless encoding of automotive lidar sensor data with very low computational complexity and high compression ratio. In this paper, we propose a solution that is based on organizing and packing lidar data into a 2-D image array and subsequently using existing image compression methods. This approach leverages image compression technology that has been developed and proven over many years of R&D, standardization, and wide deployment. In our approach, the X,Y,Z coordinates of lidar scan points are quantized, packed into one or more 2-D images, and subsequently compressed by an image codec. In addition, lidar scan points are re-ordered to optimize spatial prediction and compression efficiency. We have obtained initial results on automotive lidar data scans using several compression engines. Results using PNG and JPEG-LS and using very simple packing techniques show significant compression gains over traditional lidar data coding methods.

Introduction

Lidar sensors (a.k.a. LIDAR or LiDAR) are used widely for autonomous driving to perceive the vehicle environment. Multiple lidar sensors may be used in a single vehicle, and the resolution of lidar sensors is steadily increasing, resulting in a very large amount of data captured in each vehicle. Today, high-end lidar sensors generate on the order of a million of data points per second.

Some or all of this sensor data needs to be buffered on the vehicle, uploaded to a data center, and stored in the data center. In addition, some of the sensor data may have to be moved around within the data center or between data centers for various types of processing. The cost of such data storage and movement is steadily increasing as the amount of data increases rapidly. Hence, there is a growing need for compression of lidar sensor data.

In the case of camera (image & video) data, there is a very strong set of compression technologies available, including hardware and software, based on many years of R&D, standards development and wide deployment (JPEG, MPEG, H.26X, etc). However, this is not the case for lidar sensor data. There is a lack of compression tools for point cloud data in general, and lidar sensor data in particular. Only a few compressed formats have been developed, such as PCD with binary compression [1] and LAS/LASzip [2]. There are no broadly accepted industry standards for lidar compression, and there is undoubtedly much room to improve compression performance.

Our objective is to develop methods for compression of lidar sensor data with high compression ratio that satisfy the following requirements relevant to the autonomous driving use case:

- Lossless or near-lossless (loss is negligible for perception algorithms)
- Real-time encoding & decoding
- Computational complexity/cost must be kept very low (multiple streams handled simultaneously)
- Dynamic range of data samples is ~1 mm up to ~100/200 m, i.e. about 1:10^5
- Preserving point order is not required (i.e. permutation of points is allowed)

Approach

In this paper, we propose a solution that is based on organizing and packing the data into a 2-D image array and subsequently using existing image compression methods. Such an image-based lidar data encoding approach has several significant benefits, as follows.

1. *High compression ratio*. By packing lidar data into a 2-D image array and utilizing image compression techniques, this approach exploits 2-D spatial correlation between the samples of lidar data. This leads to a higher compression ratio, compared to other methods.
2. *Very low complexity*. Raw lidar data from automotive lidar sensors are essentially scans across 2 degrees of freedom (azimuth and elevation). In many cases, it is straightforward to pack this scan data into a 2-D grid. Subsequently, a low complexity image compression method can be used.
3. *Leverages existing image compression technology*. Image compression technology has been developed and proven over many years of R&D, standardization, and wide deployment. Hence, this approach enables quicker adoption, and reduces cost by potentially using the same compression technology for both the camera and lidar data in autonomous vehicles.

In this paper, we use *lossless* image compression and refrained from using *lossy* methods, due to the requirements for the autonomous driving application. However, the above advantages extend to the use of lossy methods for compressing lidar data as well, and indeed much higher compression ratios could be achieved. Extending further, *video* compression methods can be used to encode sequences of lidar scans. This promises again higher compression ratio, at the cost of increased complexity. In this paper, we only explored the use of still image (intra-frame) coding methods.

Prior Work

Traditionally, there exist two classes of solutions to the problem of compression of lidar data and similar types of depth/distance data.

The first class consists of solutions that directly encode raw distance data [4][7]. One disadvantage of such solutions is that the raw distance representation is strongly dependent on the specific sensor and its calibration information. This calibration information is needed to calculate the X,Y,Z coordinates in 3-D space of the sensed points at the decoder. Hence, this calibration information needs to be coded with the data. Furthermore, this approach might make the decoding process itself dependent on the specific type of sensor or application context. While this approach technically works fine in a closed environment, interoperability is reduced from a standardization and industry perspective.

The second class consists of solutions that encode the data as a point cloud. A point cloud consists of a set of points in 3-D space, each specified by its position, i.e. X,Y,Z coordinate values. For example, encoding a point cloud using an octree decomposition, or

point cloud file formats such as PCD [1] and LAS [2], fall in this class.

So far, traditional solutions in the second class address the more general problem of point cloud compression and are generally inefficient. Lidar sensor data has a specific structure due to the scanning nature of the lidar sensing process. Existing solutions in this class have not taken advantage of this structure, which allows significant improvement of compression performance.

More advanced methods have been proposed by Golla and Klein [5] and Cohen et al. [8], which consist of decomposition of a point cloud into "patches" that can be parameterized on a 2-D grid and encoding patches using height maps.

The notion of using existing (standard) image and video compression tools for encoding point cloud or lidar data has been explored previously in [4], [6], [7] and [9]. Some of these prior proposals are focused on multimedia applications and apply lossy compression tools, making the results difficult to compare. Also, most of these existing methods apply compression to range images directly. The work by Houshiar and Nuechter [6] is notable for a broad comparison of methods, including a variety of image compression tools, as well as a discussion of dynamic range of lidar data values and precision of the representation.

Currently, a subgroup within MPEG is developing a standard for point cloud compression [10] that will include image- and video-based coding tools (intra-frame and inter-frame coding); including lossy and lossless coding of geometry as well as additional point attributes (e.g. reflectance or color). Discussion of and comparison with compression tools being developed by MPEG is beyond the scope of this paper.

Proposed Method

The proposed solution for compression of lidar sensor data is based on organizing and packing the data into a 2-D image array and subsequently using an existing image compression method. In the following, we first discuss the characteristics of lidar sensor data in detail, and then describe the specific encoding/decoding steps.

Lidar Sensor Data

Although lidar data is often visualized as point clouds in a 3-D space, it is important to note that the raw measurement is simply a distance value. Lidar sensors scan the environment using one or more laser beams, for example 16, 32, or 64 beams for Velodyne sensors [12][13][14], and 4 beams for Ibeo LUX sensors [15]. Laser pulses are emitted at specific azimuth angles and elevation angles, hence each full scan can be considered parameterized in 2 dimensions, on a 2-D grid. Such a 2-D scanning grid is isomorphic to a 2-D image grid. Please note that Velodyne and Ibeo sensors are used in this paper only as examples, for purposes of technical research, and similar reasoning applies to other sensor products.

In the case of a Velodyne sensor, the raw lidar distance data for a single scan is already provided in a format that is equivalent to a 2-D array or 2-D image grid [11]. One minor issue is that the laser beams are emitted in an interleaved manner, and the data received is row-interleaved as a result. However, it is a simple matter to de-interleave "rows" in the lidar data, given the known laser beam firing sequence. Figure 1 and Figure 2 illustrate the result for a portion of a lidar scan from public Velodyne HDL-32E sensor data. For this data, the full image has a very elongated shape of 32 rows by 1846 pixels per row. Note that the figures below show only a small part of each full lidar scan.

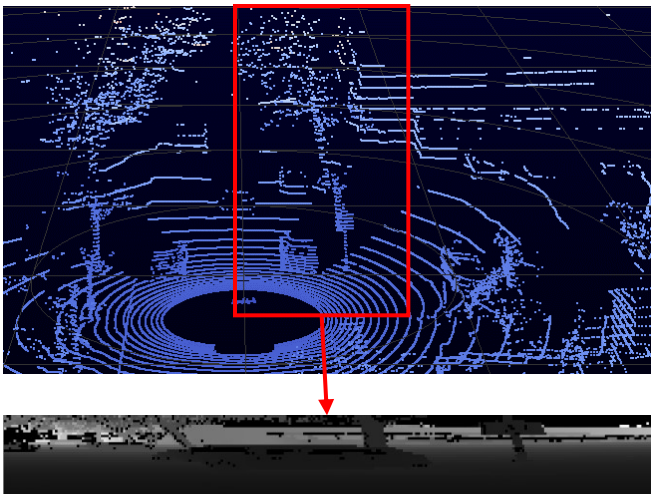


Figure 1. Portion of a 32-beam lidar scan, visualized as a point cloud in 3-D space at top, and raw distance values mapped to a 2-D image at bottom. The red rectangle highlights the approximate area of the top scan captured in the image at bottom. Public Velodyne data set.

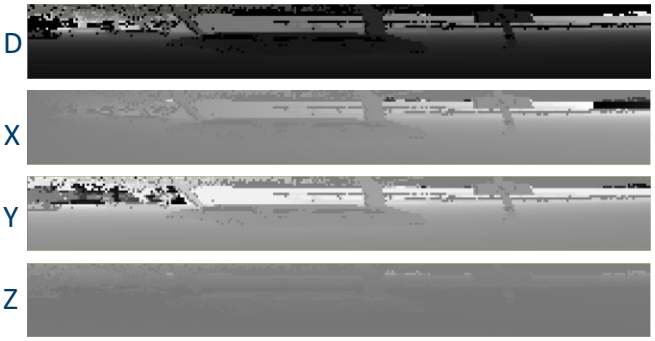


Figure 2. Portion of a lidar scan visualized in image form. The top image contains the raw distance values (identical to the image shown in Figure 1). The bottom three images show the corresponding X, Y, and Z values. These images were contrast-stretched and mapped to 8-bits to enhance visibility of the objects in the scene (the actual 16-bit values are difficult to show directly). These images contain 32 rows by 1846 columns, but in this figure the images were cropped horizontally. Derived from public Velodyne data set.

In the case of Ibeo or similar lidar sensors with smaller field of view, the typical case is that multiple sensors are used around a single vehicle, and pre-fusion of this data is applied using a hardware device. Subsequently, this pre-fused data is provided simply as a list of data points, where the original scan ordering may be lost [16]. An example of sequences of such X,Y,Z values is shown in Figure 3, and the 3-D visualization of the corresponding point cloud is shown in Figure 4. However, the original scanning pattern of e.g. 4 or 8 laser beams can be recovered from the data itself, by straightforward analysis techniques. As a result, we map lidar data originating from a distinct beam (elevation) to a distinct row in the image. In the case of this type of lidar data, each 2-D image contains only 4 or 8 rows.

Ultimately, the goal would be to pack the lidar data into one or more images such that compression efficiency is optimized. Although not explored in this paper, this can be achieved by re-ordering the scan points before encoding in a way that optimizes spatial prediction.

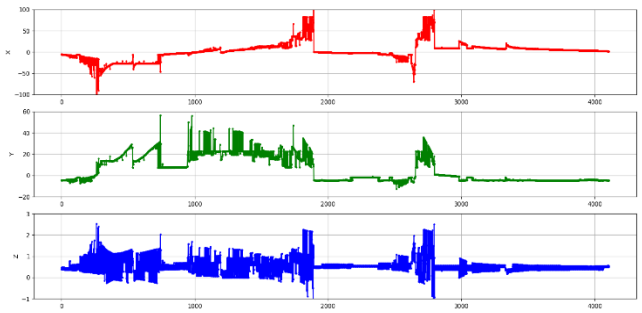


Figure 3. Example sequence of X,Y,Z values from pre-fused 4-beam lidar data (note this is a small subset of the entire sequence).

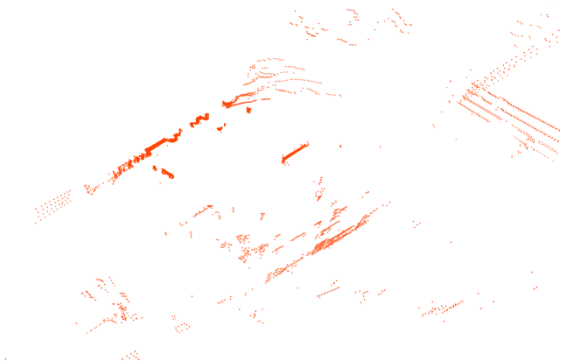


Figure 4. Point cloud visualization of the 4-beam lidar data shown in Figure 3.

It is possible that a lidar sensor does not receive a return for a specific laser emission pulse (e.g. when there is no object to reflect the pulse). In such cases, we treat this as a missing data point and simply encode it with a 0 value (for X, Y and Z, as well as distance). We treat the 0 values just as normal data values during encoding. Since such events tend to happen in entire regions of the scan (and image), the image compression engine can handle it efficiently. The effect is similar to encoding a separate occupancy map, which signals valid data points in the encoded image.

Coding individual X,Y,Z position values in fixed point representation is more efficient and precise compared to floating point representation. As mentioned above, these values have a relatively high dynamic range. Today, the precision of lidar sensors typical in autonomous vehicles is on the order of 1-2 cm [13][14][15]. The precision may improve; however a precision of 1 cm or 1 mm is likely sufficient. Conversion of floating point to fixed point values can use any suitable quantization method. In this paper, we simply applied uniform quantization across the range of possible values. As an example, to uniformly quantize X,Y,Z values to 1 cm accuracy, each individual floating point value (where 1.0 typically corresponds to 1 meter) is multiplied by 100 and subsequently rounded to the nearest integer. A suitable offset is added to handle negative values. The resulting values are represented by 16 bits (2 bytes).

Image-based Encoding/Decoding

A diagram with an overview of the proposed method is shown in Figure 5 below. The specific steps for encoding lidar sensor data are as follows:

1. Converting raw distance data from the lidar sensor to X,Y,Z point positions. This step is dependent on the specific lidar sensor and its calibration information. After conversion to X,Y,Z data, this calibration data is no longer needed. The conversion is a standard operation, taking the raw distance as well as known elevation angle and azimuth angle of the lidar beam for a lidar point as input, and computing X,Y,Z values.
2. Quantizing X,Y,Z values and converting from floating point to fixed point representation; we have used uniform quantization by scaling the values by 100.0, rounding to the nearest integer and adding a constant offset. Subsequently, each value is represented by a 16 bit value.
3. Packing the X,Y,Z values into one or more 2-D arrays; see the previous discussion for further detail.
4. Applying an existing lossless image compression/encoding method on the data in the 2-D array(s). In this paper, we applied the JPEG-LS lossless compression standard [17] and PNG encoding [18]. Both JPEG-LS and PNG can encode samples with up to 16 bits of bit-depth.

The output of the encoding steps consists of one or more compressed file(s) or bitstream(s) in standard format (e.g. JPEG-LS or PNG format). The original lidar data can be decoded simply by:

1. Applying the corresponding image decompression method.
2. Unpacking the X,Y,Z values from the decompressed 2-D image array(s).
3. Dequantizing the X,Y,Z values and converting to floating point representation.

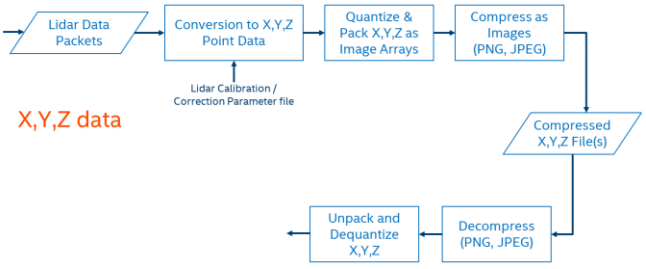


Figure 5. Overview of encoding and decoding steps using image-based compression of lidar X,Y,Z data.

Experimental Results

In our experiments, we compared the following approaches to encoding lidar scan data.

1. Using traditional point cloud or lidar formats such as PCD or LAS. Specifically, we encoded the each individual lidar scan as: a) PCD ASCII, b) PCD binary, c) PCD binary

compressed, d) LAS, e) LAZ (i.e. LASzip), and report compression efficiency for each.

2. Proposed approach as shown in Figure 5, where data is converted to X,Y,Z values, quantized, packed into image arrays and then compressed as a *single 3-channel image* (single image with XYZ channels treated as RGB channels). We report compression efficiency for: a) the uncompressed PPM image file, b) applying Linux gzip to the PPM file, c) PNG compression, d) JPEG-LS compression, and e) ZFP compression.
3. Proposed approach as shown in Figure 5, where data is converted to X,Y,Z values, quantized, packed into image arrays and then compressed separately as *three 1-channel images* (similar to grayscale images). We report compression efficiency for: a) the three uncompressed PGM image files, b) applying Linux gzip to the PGM files, c) PNG compression, d) JPEG-LS compression, and e) ZFP compression.
4. Alternative approach shown in Figure 6, where raw lidar distance values are directly quantized and packed into a single image array, and then compressed as a 1-channel image. Again, we report compression efficiency for: a) the uncompressed PGM image file, b) applying Linux gzip to the PGM file, c) PNG compression, d) JPEG-LS compression.

ZFP is a method proposed for compression of floating point arrays [3], which we applied after scaling and quantization of floating point X,Y,Z or distance values, to the same precision used for our image compression results. The results for LAS and LASzip are based on using the same precision (0.01 m).

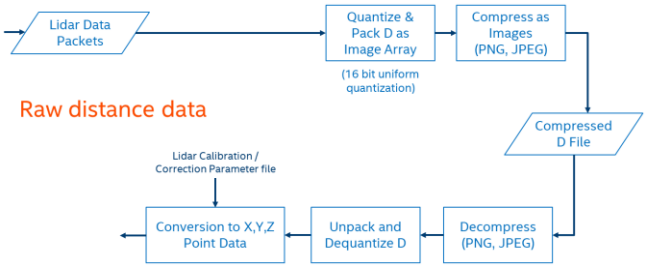


Figure 6. Overview of encoding and decoding steps using image-based compression of lidar distance (range) data.

The main compression efficiency metric used is the average number of bytes per point, which is simply computed from the file size (in bytes) and dividing it by the number of points in a lidar scan. Subsequently, the average is taken over all the scans in a sequence. We don't express the result as a compression ratio, since there is not a well-established reference. No distortion metric is used since only lossless compression is applied. All our results only consider the geometry data (X,Y,Z coordinates or distance/range data), excluding other attributes such as reflectance or color.

Experimental results were obtained using data sets from both Velodyne [13] and Ibeo [15] lidar sensors. Again, please note that this data is only used as example data, and used only for research purposes. Two public Velodyne data sets were used, captured with the HDL-32E sensor, from a moving vehicle, each 100 scans over 10 sec (10 scans/sec), about 59,000 points per scan. The first is referred to as the "Tunnel" lidar stream, the second is referred to as the "Highway" lidar stream. For these two streams, the average number of bytes per point for all methods listed is shown in Figure 7 and Figure 8 respectively.

Figure 9 shows average number of bytes per point for a subset of methods, applied to a lidar stream captured from a vehicle using Ibeo Lux 4L lidar sensors. The data consisted of 600 scans captured over 24 sec (25 scans/sec), about 4,000 points per scan. In the vehicle setup, data from 6 individual sensors were pre-fused by a hardware fusion device.

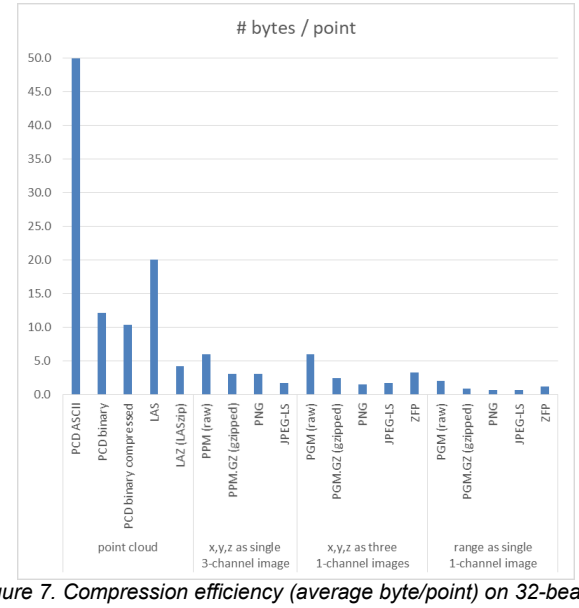


Figure 7. Compression efficiency (average byte/point) on 32-beam "Tunnel" lidar data.

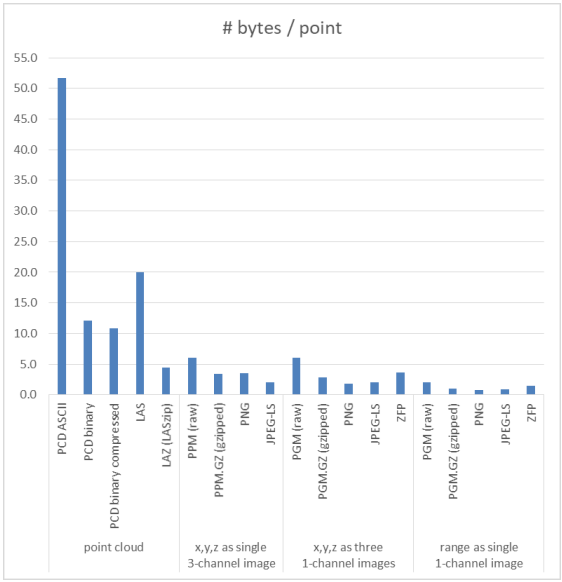


Figure 8. Compression efficiency (average byte/point) on 32-beam "Highway" lidar data.

While the ASCII PCD format is clearly very inefficient, PCD binary format and PCD binary compressed format reach to about

10 bytes/point. LAS itself encodes at a fixed 20 bytes/point as expected, while LASzip can reach under 5 bytes/point.
Image-based compression is generally more efficient, as can be seen in the results. Raw PGM or PPM files use about 6 bytes/point as expected (2 bytes for X, for Y, and for Z). Image-based compression with PNG and JPEG-LS is generally the most efficient in these results, approximately 2 bytes/point or less.
These results indicate that there is no advantage in encoding the X, Y, and Z images as a single 3-channel image as compared to encoding as three separate 1-channel images. The results appear equivalent when using JPEG-LS. In fact, when using PNG, the results appear worse when encoding as a single 3-channel image.
As shown in Figure 7 and Figure 8, encoding the data as a single range/distance image always results in the smallest file size, and is more efficient than encoding three X, Y and Z images. As discussed, the trade-off is higher complexity at the decoder.

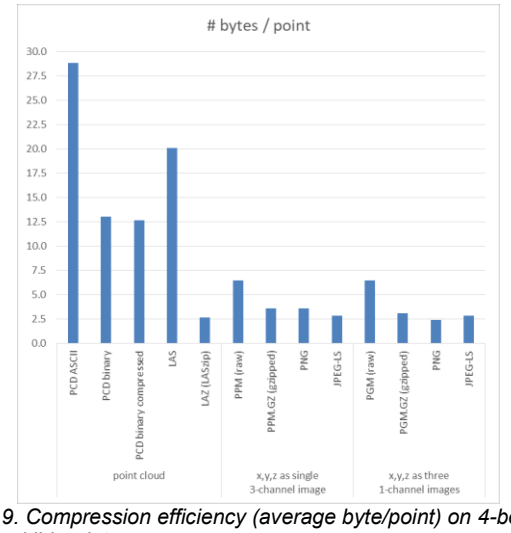


Figure 9. Compression efficiency (average byte/point) on 4-beam pre-fused lidar data.

The tables below show the same results in numerical form. An additional column is included showing the compressed data rate (in Mbit/s) for information. Table 1 and Table 2 shows results taken with a vehicle-mounted Velodyne HDL-32E sensor. Table 3 shows results on lidar data from vehicle-mounted Ibeo 4L sensors.

Table 1. Compression efficiency (average byte/point) on 32-beam "Tunnel" lidar data.

geometry packing	compression type	file size (bytes)	byte/point	Mbit/s
point cloud	PCD ASCII	2887908.2	49.9	231.0
	PCD binary	705706.2	12.2	56.5
	PCD binary compressed	602628.6	10.4	48.2
	LAS	1169577.4	20.0	93.6
	LAZ (LASzip)	243527.7	4.2	19.5
x,y,z as three 1-channel images	PGM (raw)	350856.1	6.1	28.1
	PGM.GZ (gzipped)	140470.0	2.4	11.2
	PNG	**90096.8**	**1.6**	**7.2**
	JPEG-LS	**98897.7**	**1.7**	**7.9**
	ZFP	188589.4	3.3	15.1
range as single 1-channel image	PGM (raw)	116952.0	2.0	9.4
	PGM.GZ (gzipped)	52713.0	0.9	4.2
	PNG	**39016.2**	**0.7**	**3.1**
	JPEG-LS	**40985.3**	**0.7**	**3.3**
	ZFP	73619.4	1.3	5.9

Table 2. Compression efficiency (average byte/point) on 32-beam "Highway" lidar data.

geometry packing	compression file type	file size (bytes)	byte/point	Mbit/s
point cloud	PCD ASCII	3045411.7	51.6	243.6
	PCD binary	711881.0	12.1	57.0
	PCD binary compressed	636188.6	10.8	50.9
	LAS	1179868.6	20.0	94.4
	LAZ (LASzip)	264164.7	4.5	21.1
x,y,z as three 1-channel images	PGM (raw)	353943.5	6.0	28.3
	PGM.GZ (gzipped)	165216.7	2.8	13.2
	PNG	**109239.1**	**1.9**	**8.7**
	JPEG-LS	**122033.6**	**2.1**	**9.8**
	ZFP	217791.7	3.7	17.4
range as single 1-channel image	PGM (raw)	117981.2	2.0	9.4
	PGM.GZ (gzipped)	59114.0	1.0	4.7
	PNG	**45006.7**	**0.8**	**3.6**
	JPEG-LS	**49631.0**	**0.8**	**4.0**
	ZFP	84522.7	1.4	6.8

Table 3. Compression efficiency (average byte/point) on pre-fused 4-beam lidar data.

geometry packing	compression type	file size (bytes)	byte/point	Mbit/s
point cloud	PCD ASCII	117336.3	28.8	9.4
	PCD binary	52969.7	13.0	4.2
	PCD binary compressed	51511.4	12.6	4.1
	LAS	81683.2	20.1	6.5
	LAZ (LASzip)	**10873.3**	**2.7**	**0.9**
x,y,z as three 1-channel images	PGM (raw)	26469.9	6.5	2.1
	PGM.GZ (gzipped)	12634.7	3.1	1.0
	PNG	**9936.4**	**2.4**	**0.8**
	JPEG-LS	**11603.3**	**2.8**	**0.9**
x,y,z as single 3-channel image	PPM (raw)	26437.9	6.5	2.1
	PPM.GZ (gzipped)	14643.0	3.6	1.2
	PNG	**14712.8**	**3.6**	**1.2**
	JPEG-LS	**11543.3**	**2.8**	**0.9**

Conclusions

In this paper, we explored application of existing standard image compression technology to lidar scan data, for automated driving applications. Using proven image compression technology for lidar data compression potentially reduces risk and development time, compared to developing a new compression method from scratch. Another benefit is enabling the use of the same compression technology for both image and lidar data in the context of automated driving applications, which involve capture and storage of large amounts of image and video data in addition to lidar data.
We have shown how raw lidar scan data can be packed into a 2-D image format with very low complexity. We have applied several lossless image compression engines to the re-organized lidar data. Our initial lossless compression results on lidar data are quite promising, in particular when using JPEG-LS. These results were obtained using very simple methods to pack the lidar data into 2-D arrays and with very little optimization. We expect an increase in compression efficiency can be achieved by packing the data using better methods. Additional significant gains in compression efficiency can be achieved by using video compression techniques, as well as using lossy compression techniques (in a way where the fidelity can be tightly controlled). Finally, a performance comparison should be made to the upcoming MPEG Point Cloud Compression standard insofar it can support the automated driving application requirements.

References

[1] The PCD (Point Cloud Data) file format, http://pointclouds.org/documentation/tutorials/pcd_file_format.php

[2] M. Isenburg, "LASzip: lossless compression of LiDAR data," European Lidar Mapping Forum (ELMF), Salzburg, Austria, 2011.

[3] P. Lindstrom, "Fixed Rate Compressed Floating-Point Arrays," IEEE Trans. on Visualization and Comp. Graphics, vol. 20, no. 12, Dec. 2014.

[4] F. Nenci et al., "Effective Compression of Range Data Streams for Remote Robot Operations using H.264," Int. Conf. on Intelligent Robotics and Systems (IROS), Chicago, IL, USA, 2014.

[5] T. Golla and R. Klein, "Real-time Point Cloud Compression," Int. Conf. on Intelligent Robotics and Systems (IROS), Hamburg, Germany, 2015.

[6] H. Houshiar and A. Nuechter, "3D Point Cloud Compression using Conventional Image Compression for Efficient Data Transmission," Int. Conf. on Information, Communication and Automation Technologies (ICAT), Sarajevo, Bosnia and Herzegovina, 2015.

[7] C. Tu et al., "Compressing Continuous Point Cloud Data Using Image Compression," Int. Conf. on Intelligent Transportation Systems (ITSC), Rio de Janeiro, Brazil, 2016.

[8] R. Cohen et al, "Compression of 3-D point clouds using hierarchical patch fitting," Int. Conf. on Image Processing (ICIP), Beijing, China, 2017.

[9] R. Mekuria et al., "Design, Implementation, and Evaluation of a Point Cloud Codec for Tele-Immersive Video," IEEE Trans. on Cir. and Systems for Video Technology, vol. 27, no. 4, April 2017.

[10] MPEG, Point Cloud Compression, https://mpeg.chiariglione.org/standards/mpeg-i/point-cloud-compression.

[11] Velodyne, "Application Note, HDL-32E: Packet Structure & Timing Definition," https://velodynelidar.com/downloads.html.

[12] Velodyne, "VLP-16 Lidar Sensor Datasheet," https://velodynelidar.com/downloads.html.

[13] Velodyne, "HDL-32E Lidar Sensor Datasheet," https://velodynelidar.com/downloads.html.

[14] Velodyne, "HDL-64E S3 Lidar Sensor Datasheet," https://velodynelidar.com/downloads.html.

[15] Ibeo automotive, "Ibeo LUX 4L / LUX 8L / LUX HD Data Sheet", 2017.

[16] Ibeo automotive, "Interface Specification for ibeo LUX, ibeo LUX systems and ibeo Evaluation Suite," 2017.

[17] M. Weinberger et al., "The LOCO-I Lossless Image Compression Algorithm: Principles and Standardization into JPEG-LS," IEEE Trans. on Image Processing, 2000.

[18] W3C, "Portable Network Graphics (PNG) Specification (Second Edition)," ISO/IEC 15948:2003 (E), W3C Recommendation, November 2003.

Author Biography

Peter van Beek is with Intel's Silicon Valley Innovation Center, supporting development of Mobileye's solutions for safe and scalable autonomous driving. Before joining Intel, Peter was at Sharp Labs of America, developing image/video/vision technology for mobile robot and security camera applications; 4K TV applications; automated visual inspection; video streaming; and multimedia applications. Peter received M.Sc.Eng and Ph.D. degrees in Electrical Engineering from the Delft University of Technology, the Netherlands.

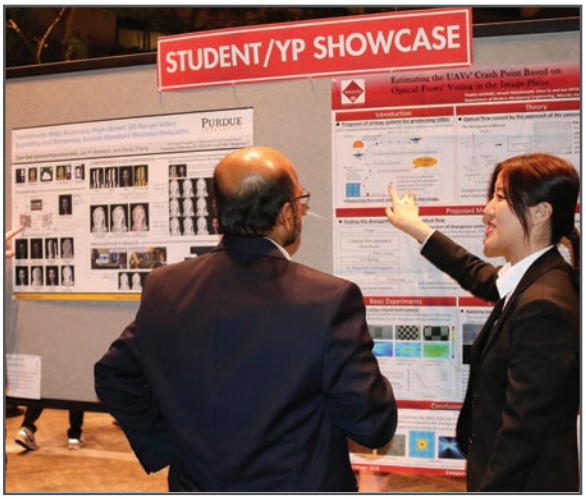
JOIN US AT THE NEXT EI!

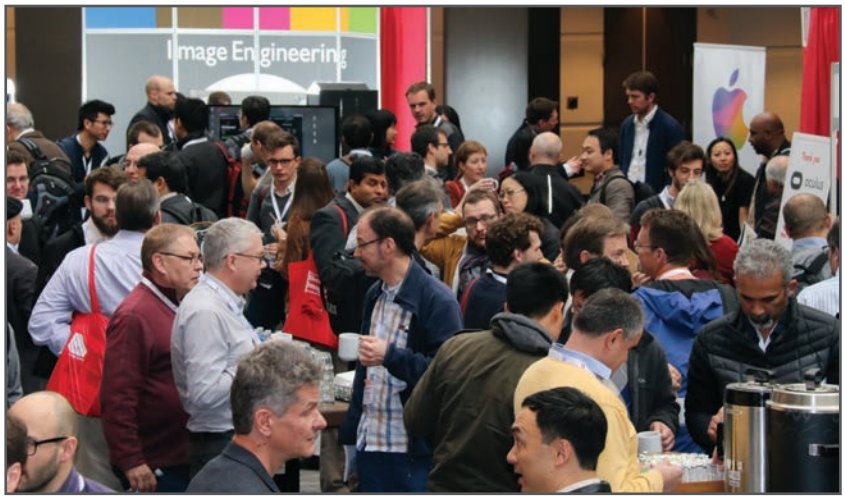
IS&T International Symposium on

Electronic Imaging

SCIENCE AND TECHNOLOGY

Imaging across applications . . . Where industry and academia meet!





• SHORT COURSES • EXHIBITS • DEMONSTRATION SESSION • PLENARY TALKS •
• INTERACTIVE PAPER SESSION • SPECIAL EVENTS • TECHNICAL SESSIONS •

www.electronicimaging.org

IS&T
imaging.org