RIDDLE: Lidar Data Compression with Range Image Deep Delta Encoding Supplementary Material

Xuanyu Zhou* Charles R. Qi* Yin Zhou Dragomir Anguelov
Waymo LLC

A. Overview

In this supplementary, we provide more details of our method, extra analysis experiment results and visualizations. In Sec. B, we describe more details of the deep predictive model, including its network architecture, losses and its training process as well as more explanation of the input data transformations. In Sec. C, we provide more analysis experiment results on model latency, effects of the entropy encoder choices and effects of the context sizes. In Sec. D, we apply our method to compress lidar data attributes beyond the range values. Finally, in Sec. E, we provide more visualizations of our model's predictions.

B. Details of the Predictive Models

Model architecture For deep prediction model, we adapt the structure of PointNet [6]. Details of layers are visualized in Fig. 1. To reduce the latency, the network channel sizes are halved compared to the original architecture and the T-Nets are removed. After concatenation of global and local features, we split the network into anchor-classification branch and residuals branch. Each branch is a MLP with layer sizes [128, 64, # of anchors], where # of anchors is 99 for intra-prediction model and 199 for temporal model. For temporal model, We build the KDtrees using neighbors.NearestNeighbors method by Scikit-learn library. We use the left valid depth and up valid depth as estimates to each query 50 points from the last frame as the temporal context.

Loss functions At training time, we train deep prediction model end-to-end with the anchor classification and the anchor residual regression loss. We weight the classification loss by a weight.

$$\mathcal{L} = \gamma \mathcal{L}_{\text{classification}} + \mathcal{L}_{\text{regression}} \tag{1}$$

The classification loss is a cross-entropy loss across $hw - 1$ classes for intra-frame prediction model and $hw - 1 + m$ classes for temporal model. The ground truth class is selected as the index of the pixel with the closest distance to the to-be-predicted pixel. As the input are quantized values, there could be ties. To avoid ties we add a bias term to the distances to favor the pixels that are closer in angles (absolute delta azimuth + absolute delta elevation) to the to-be-predicted pixel. The regression loss is a L1 loss between the predicted residual corresponding to the pixel of the ground truth anchor and the ground truth residual of that pixel.

Training We learn the weights of the prediction model by training on the range images from the Waymo Open Dataset train set. We randomly crop the patches of shape 10×10 from the range images and train the deep network with batch size 128 and an Adam optimizer. We use loss weight $\gamma = 0.01$. The initial learning rate is 0.00005, and we decay learning rate by 10x at step 1500k and step 3000k. We normalize the range values to $[0, 1]$ by dividing them by 75m. To mimic the same setting in decoding, the training inputs are quantized. The ground truth attribute values are in full precision for more accurate supervision. For pixel locations at the boundary of the range images, we enforce the same patch size via zero padding.

There are two strategies for inputs quantization. The first strategy is to train different models for different quantization precisions, and for each model we use a fixed quantization precision for the input. The second strategy is using mixed precisions to quantize the input during training. Specifically, we uniformly sample a quantization precision for a given input from 0.0001 to 0.5000 with sample bin size of 0.0001. By the second strategy, we only need to train one model for different compression rates. From our experiments, we observe that the second strategy won't harm the compression rates at individual quantization precision.

Baselines For the previous valid value method, we predict the attribute $I_{i,j}$ at row i, column j to be $\hat{I}_{i,j} = I'_{i,j-1}$, if $I'_{i,j-1}$ is a valid pixel (not void due to empty laser return) else repeatedly decrement j by 1 until the pixel is valid. For the linear interpolation baseline method, we predict $\hat{I}_{i,j} =$

*equal contribution

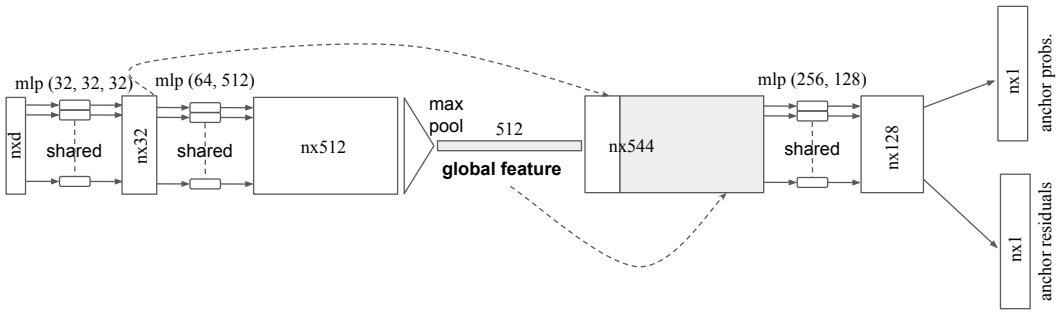


Figure 1. **Deep predictive model architecture.** n is the number of context points, d is the input point dimension ($d = 3$ for the intra-prediction model, $d = 4$ for the temporal model).

$I'_{i,j-1} + I'_{i-1,j} - I'_{i-1,j-1}$. For the 12-layer CNN method, we adapt a similar structure as ResNet [3]. The network is composed of two convolutional layers (channel sizes 64, 32) and 5 residual blocks (channel sizes 32, 32 for each block), and all convolutional layers have filter size 3x3.

C. More Analysis

Analysis of the model latency. Table 3 shows the latency comparison between our method and OctSqueeze [4] and G-PCC from MPEG. To achieve faster decompression speed, during compression, we split a range image into smaller blocks and run compression in parallel on these blocks. During decompression, we decode in parallel on these smaller blocks. Our experiments show that if we split a 64 by 2650 range image into 212 blocks with size 16 by 50, the bitrate would only increase by 0.5%, which is nearly negligible. If we split it into 424 blocks with size 16 by 25, the bitrate would increase by 5.13%. Table 3 shows the latency of our method by splitting into blocks with size 16 by 26 during decoding. We benchmark our method on NVIDIA Tesla V100 GPU. Our deep model is accelerated by TensorRT with float16 quantization. Operations (entropy encoding) other than model inference is written in C++. The latency of G-PCC is benchmarked on CPU using MPEG's implementation (github.com/MPEGGroup/mpeg-pcc-tmc13). The latency of OctSqueeze (depth 16) is from the original paper [4]. Moreover, we believe further speedup of our method could be achieved by methods like predicting multiple pixels at a time or shared point embedding for streamed prediction.

Choices of entropy encoder After the predictive delta encoding, we get a residual map of the range image. An entropy encoder is used to leverage the sparsity pattern in the residual map to compress it. Given an accurate prediction model, most of the residuals would be zero. In addition, as shown in Fig. 2, larger quantization steps would round more residuals to zero, thus the residuals would become more sparse. We adapted two methods to entropy encode the residuals. In practice, we can select the entropy encoder with the highest compression rates depending on the quantization rates and the predictor.

entropy encoder	quantization	bpp
sparse repre. + arithmetic encoding	0.1m	**2.28**
varints+LZMA	0.1m	2.33
huffman encoding	0.1m	2.49
arithmetic encoding	0.1m	2.41
sparse repre. + arithmetic encoding	0.02m	4.17
varints+LZMA	0.02m	**4.08**
huffman encoding	0.02m	4.25
arithmetic encoding	0.02m	4.21

Table 1. **Ablation study of the entropy encoders.**

The first method is to represent the residuals using sparse representation. Given an array of residuals, we represent the array with the values of nonzero residuals and their indices in the array. For a long run of sparse residuals, the sparse representation would be quite memory efficient. After obtaining the sparse representation of residuals, we use arithmetic encoding to further reduce its size.

The second method is to represent the residuals using run-length encoding. We first flatten the residual map to a vector and then represent it with the values and the run-length of values. This representation achieves better compression rates when the residuals are not that sparse, i.e. when quantization step size is small. After obtaining the run-length representation, we use LZMA compressor to further reduce its size.

Table. 1 shows that different entropy encoders have different compression rates of the residuals. For quantization precision of 0.1m, the residuals are more sparse, and the compression rate of using sparse representation (representing non-zero residuals by specifying their row, col index and the residual values) with arithmetic encoding is higher than varints with LZMA. However, for quantization precision of 0.02m, the compression rate of varints with LZMA

method	compressing (ms)	decompressing (ms)
G-PCC [2]	1594.5	1052.1
OctSqueeze [4]	106.0	902.3
RIDDLE (ours)	532.51	966.3

Table 2. **Latencies of lidar data compression methods.** Note the G-PCC is evaluated using CPU on the Waymo Open Dataset (WOD). OctSqueeze is evaluated on the KITTI dataset (with a similar range image resolution to WOD) and our method is evaluated on the WOD. Both OctSqueeze and our method use GPU for model inference.

preprocessing (ms)	network (ms)	entropy encoding (ms)
16.23	487.4	28.88

Table 3. **Breakdown of encoding time of RIDDLE.** Preprocessing includes the time to compute and compress a binary mask indicating whether a pixel is a valid return in the range image. Network refers to the model prediction time. Entropy encoding refers to the time used by entropy encoder.

is higher, due to the decrease of zero residuals.

Analysis on input choices. Table 4 shows how the input choices affect the prediction accuracy. Instead of inputing intra-frame context of 10 by 10 (minus the bottom right one), we can just input the up 9 pixels (row 1), the right 9 pixels (row 2) or smaller context size (row 4). We can see that enlarging the receptive field of input with context from both upper left and upper right can improve the prediction accuracy (row 3 v.s. row 1 and 2). Moreover, including azimuth and inclination as additional input attributes can also improve the predictor (row 4 v.s. row 3) compared to just using the relative row/column indices as input.

Generalization of the method. When we apply the deep predictive model trained on 64-beam frames on the Waymo Open Dataset directly to the subsampled 32-beam frames, it achieves 2.55 bpp at 0.1m depth precision (only slightly larger than 2.23 bpp on 64-beams). In addition, the compressor trained on WOD can apply well in KITTI (Fig. 5), which shows the generalization of it.

Comparison with LASzip. Benchmarked on Waymo Open Dataset, LASzip [5] has 67.6 PSNR and 0.0048 Chamfer Distance at 10.62 bpp. Our method achieves 72.39 PSNR and 0.0026 Chamfer Distance at 4.51 bpp, which clearly outperforms it.

context size	input format	acc. @0.1m
10 x 1	(Δazimuth, Δinclination, depth)	37.53
1 x 10	(Δazimuth, Δinclination, depth)	60.00
5 x 10	(Δrow index, Δcol index, depth)	65.02
5 x 10	(Δazimuth, Δinclination, depth)	65.21
10 x 10	(Δazimuth, Δinclination, depth)	**65.75**

Table 4. **Effects of context size and input format.** We used the intra-frame model for this evaluation.

D. Compression of More Attributes

Since a lidar point cloud may contain additional attributes (e.g. intensity, elongation) than the range values, in this section we show how much we can compress the other attributes than ranges. We train a network to take in multi-channel range images and output multi-channel prediction. Specifically, we train a network on the Waymo Open dataset, which contains 3 channels for each point: range, intensity and elongation. The network is modified to have 3 anchor-classification branches and 3 residuals branches for 3 attributes. From Table 5 row 1-4, we can see that a quantization precision 0.02m for range, or 0.1 for intensity, or 0.1 for elongation have similiar effect on the object detector. With those quantization precisions for each attributes, range values account for most of the storage cost (4.04 bpp) compared to the other two (0.88 bpp for intensity and 0.78 bpp for elongation).

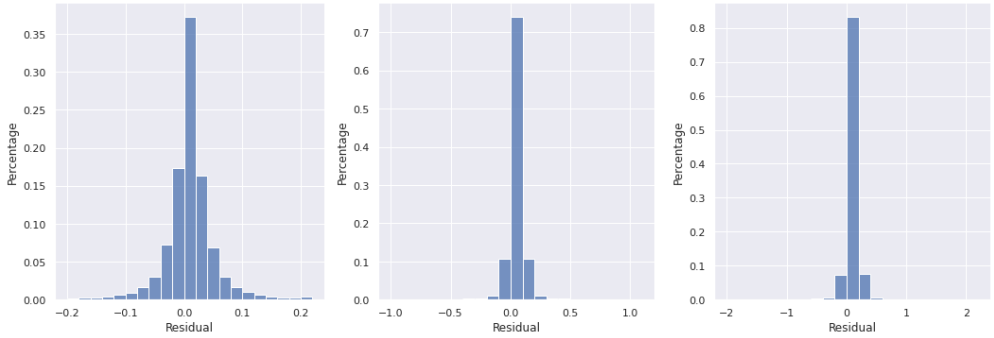
E. More Visualizations

The distributions of residuals Fig. 2 shows the distribution of the range residual maps after our deep delta encoding step. We can see that the larger the quantization interval the more concentrated are the residuals (lower entropy), which explains the lower bitrate after the compression. Note that for a quantization size of 0.1m, more than 70% of the prediction has zero error compared to the ground truth quantized range image.

Auto-encoder-based compression We further compare our method with an auto-encoder-based image compression algorithm [1]. $r = \max_{p_i \in P} \min_{j \neq i} \|p_i - p_j\|_2$. The auto-encoder is trained with a learning rate of 0.0001 and an Adam optimizer. The range values are scaled to [0, 1] by 75m instead of by 255 as for RGB images. Fig. 3 shows the reconstructed point clouds of the auto-encoder-based method and our method under similar bitrates. The colors of points in the visualizations demonstrate that our method has much better reconstruction quality compared to this auto-encoder baseline. The auto-encoder-based range image compression method does poorly especially at the boundary between foreground points and background points.

precision			bit per point			total bpp	vehicle mAP	pedestrian mAP
range	intensity	elongation	range	intensity	elongation			
-	-	-	32	32	32	96.00	69.59	65.62
0.02m	-	-	4.04	32	32	68.04	69.60	65.63
-	0.1	-	32	0.88	32	64.88	69.59	65.84
-	-	0.1	32	32	0.78	64.78	69.59	65.62
0.02m	0.1	0.1	4.04	0.88	0.78	5.70	69.59	65.62

Table 5. **Effects of context size and input format.** The uncompressed attribute is saved as 32-bit float numbers. We that quantizing the intensity or elongation to 0.1 or quantizing the range value to 0.02m has little impact on the detection mAPs. At these selected quantization rates, the range channel accounts for most of the bpp (around 70%).



(a) Residuals distribution (0.02m) (b) Residuals distribution (0.1m) (c) Residuals distribution (0.2m)

Figure 2. **Distribution of residuals at different quantization precisions**. Proposed deep prediction model is able to accurately model the joint distributions of range image pixel attributes, resulting in a concentrated distribution of residuals with low entropy.

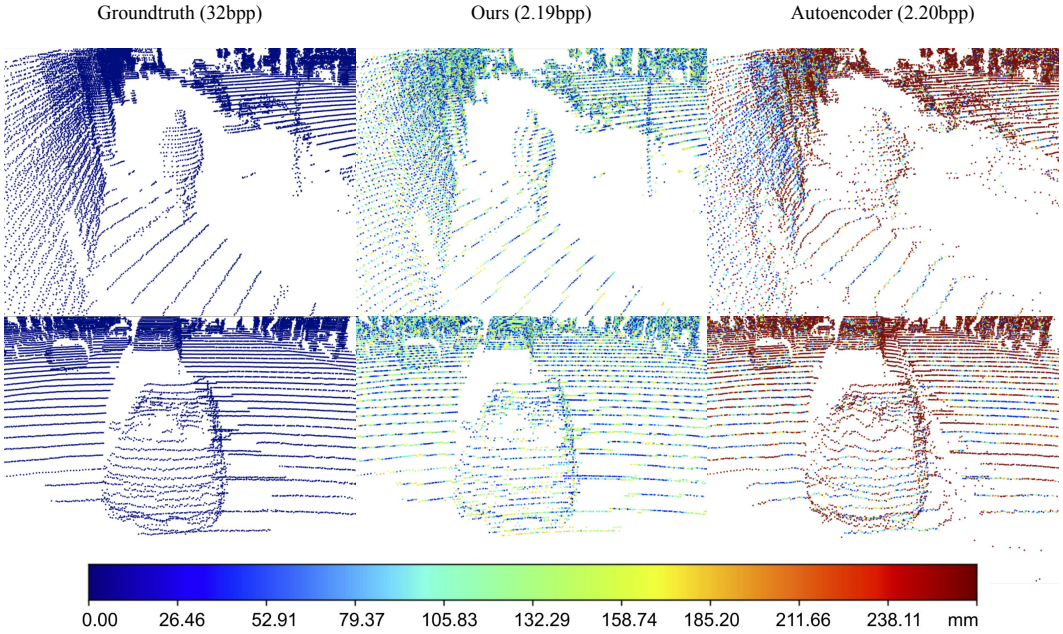


Figure 3. **Visualization of reconstructed point clouds, colored by per point Chamfer distance** (error bar colormap on the bottom). From left to right: raw, RIDDLE (ours) and auto-encoder. It is clear that our method, under the same bit per point, has mush less distortion. Best viewed in color with zoom in.

References

[1] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. *arXiv preprint arXiv:1802.01436*, 2018. 3

[2] D Graziosi, O Nakagami, S Kuma, A Zaghetto, T Suzuki, and A Tabatabai. An overview of ongoing point cloud compression standardization activities: video-based (v-pcc) and geometry-based (g-pcc). *APSIPA Transactions on Signal and Information Processing*, 9, 2020. 3

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2

[4] Lila Huang, Shenlong Wang, Kelvin Wong, Jerry Liu, and Raquel Urtasun. Octsqueeze: Octree-structured entropy model for lidar compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1313–1323, 2020. 2, 3

[5] Martin Isenburg. Laszip: lossless compression of lidar data. *Photogrammetric Engineering Remote Sensing*, 79, 02 2013. 3

[6] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 1