Point Cloud Compression with Range Image-based Entropy Model for Autonomous Driving

Sukai Wang[1,4] and Ming Liu[1,2,3]

[1] The Hong Kong University of Science and Technology, Hong Kong SAR, China
[2] The Hong Kong University of Science and Technology (Guangzhou), Nansha, Guangzhou, 511400, Guangdong, China
[3] HKUST Shenzhen-Hong Kong Collaborative Innovation Research Institute, Futian, Shenzhen
[4] Clear Water Bay Institute of Autonomous Driving, Hong Kong SAR, China
{swangcy, eelium}@ust.hk

Abstract. In this supplementary material, we first introduce the network details and the training details, and then further evaluate and discuss the runtime of our compression and decompression framework. We also visualize the comparison results of our method and the baseline methods with the error bar in three different datasets (KITTI, Oxford, and Campus16) and exhibit the detection performance in the KITTI detection dataset.

1 Compression Framework

Our proposed framework is the first to apply the entropy model in point cloud compression based on the range image. And we find that the neural network is suitable for the coarse range image prediction and refinement. Compared with the RGB image compression technologies, our proposed framework doesn't alter the original data and achieves better performance using the geometry features. From the experiments of the SOTA OctSqueeze[2], we can find that directly applying the image deep neural network into the range image cannot get good enough results, even worse than the G-PCC and the other traditional baselines. Also, currently, the SOTA tree-based learnable networks, like OctSqueeze[2] and VoxelContext[4] are not open-source for reproduction. We are promised to release our code for better usage and comparison.

2 Network Details

The RICNet$_{stage1}$ and RICNet$_{stage2}$ contain a Minkowski Unet14 architecture as the 3D feature extractor, two scan-attentive feature conv (SAC) blocks as the 2D feature fusion module, and one 1D conv layer as the occupancy head or refinement head for two sub-models respectively. The detailed overall architecture of the RICNet is shown in Fig. 1.

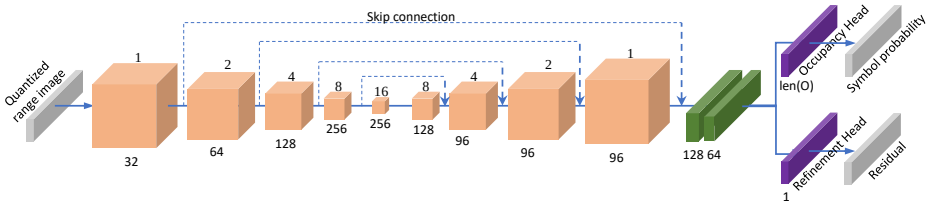

Fig. 1. The overall architecture of our proposed RICNet. The orange blocks are the Minkowski Conv blocks and TransConv blocks [1], the dot lines are the skip connections, green blocks are scan-attentive feature conv blocks, and purple blocks are the occupancy head and the refinement head for RICNet$_{stage1}$ and RICNet$_{stage2}$ respectively. The numbers below the blocks mean the number of the output channels, and the numbers on the top of the blocks represent the global stride with the original block in 3D convolution.

3 Training Details

The proposed RICNet is implemented on Pytorch and trained on an Intel 3.7GHz i7 CPU and a single GeForce GTX 1080Ti graphics card. We use an Adam [3] optimizer for training, in which beta1=0.9, beta2=0.999, epsilon=1e-08. The cyclical learning rates (CLR) [5] is applied for a higher training speed. In training, the batch size is 2, and the base learning rate is 5e-4 with the cycle range as one epoch.

4 Runtime Evaluation

In this section, we evaluate the runtime of each step of our proposed three-stage compression and decompression framework, and the results are shown in Tab. 1. From the results, we can find that using the non-ground points in RICNet$_{stage1}$ can save 0.06s for probability prediction.

One of our method's drawbacks is that our method cannot be implemented in real-time. If we need the speed of our compressor more than its compression rate, we can choose to drop stage 0 and stage 1 in our proposed framework and use a basic compressor (BZiP2) to encode the quantized range image from Q_2. We have tested that when we use a single basic compressor to replace the stage 0 and stage 1 (compress the quantized point cloud with q_2), the runtime can be reduced to 0.07 for encoding and 0.03 for decoding, though the BPP will drop by about 0.3 (18%).

5 Additional Qualitative Results

In this section, we visualize the comparison results of our method and the baseline methods with the error bar in three different datasets (KITTI with Velodyne HDL-64, Oxford with Velodyne HDL-32, and Campus with Velodyne VLP-16)

stage	compression					decompression				
	stage 0		stage 1		stage 2	stage 0		stage 1		stage 2
	RS	bc end	net	ac enc	net	RS	bc dec	net	ac dec	net
time/s	0.006	0.21	0.17	0.013	0.23	0.006	0.015	0.17	0.009	0.23

Table 1. The runtime of each step in our proposed three-stage compression and decompression framework. *RS* means the RANSAC used in ground extraction and segmentation in stage 0, *bc enc* and *bc dec* are the encoding and decoding time for the basic compressor (BZiP2), *net* in stage 1 and stage 2 mean the RICNet$_{stage1}$ and RICNet$_{stage2}$ respectively, *ac enc* and *ac dec* are the encoding and decoding time for the arithmetic coding algorithm.

in Fig. 2, and exhibit the detection performance of the KITTI detection dataset in a multi-car environment in Fig. 3 and multi-pedestrian environment in Fig. 4. The BPP, compression ratio and chamfer distance results of each row (method) in these three figures are in the corresponding position of the attached table.

In this additional qualitative experiments, the settings of our method:

$$q_1 = 0.3, q_2 = 0.1,$$

and the settings of the baseline algorithms:

- Draco:
$$qp = 10, cl = 7;$$
- G-PCC:
$$Accuracy = 0.07;$$
- R-PCC (Uniform):
$$Accuracy = 0.1.$$

From the results, we can see that our proposed method outperforms the state-of-the-art baseline methods in terms of the reconstruction quality and downstream detection performance dramatically.

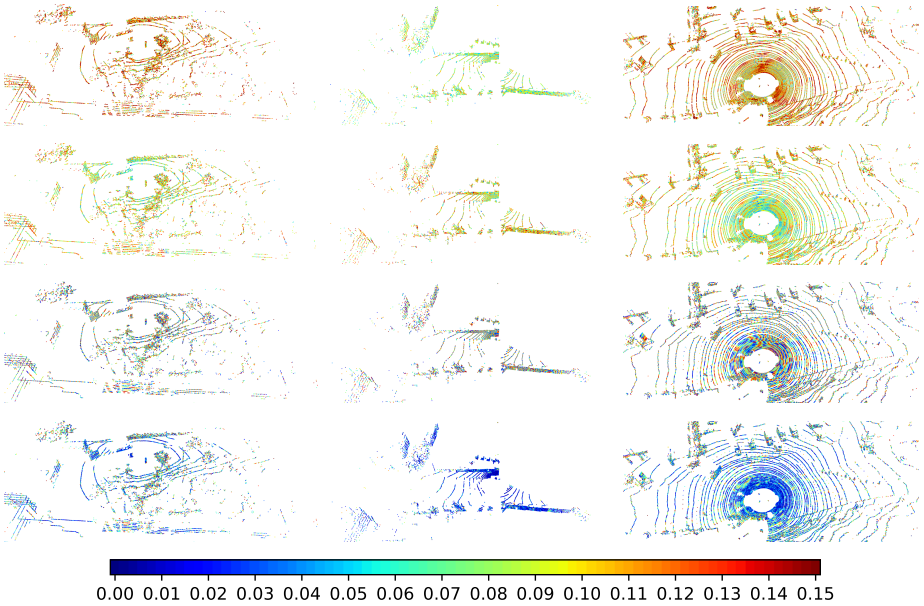

Fig. 2. The qualitative results of the reconstructed point cloud with the original point cloud. From left to right, there are three datasets: Campus with Velodyne VLP-16, Oxford with Velodyne HDL-32, and KITTI with Velodyne HDL-64. From top to bottom, there are four comparative algorithms: Draco, G-PCC, R-PCC, and Ours. The BPP, compression ratio and chamfer distance results of each subfigure are in the corresponding position of Tab. 2.

	Campus	Oxford	KITTI
Draco	5.15 / 18.62 / 0.071	5.69 / 16.87 / 0.050	2.83 / 33.83 / 0.065
G-PCC	5.42 / 17.71 / 0.060	5.43 / 17.68 / 0.061	3.21 / 29.88 / 0.058
R-PCC	4.18 / 22.92 / 0.048	4.36 / 22.0 / 0.047	2.12 / 45.25 / 0.047
Ours	**4.01 / 23.94 / 0.033**	**4.03 / 23.82 / 0.014**	**1.97 / 48.73 / 0.028**

Table 2. The BPP ↓ / compression rate ↑ / mean chamfer distance (m)↓ of each corresponding subfigure in Fig. 2 (corresponding rows and cols).

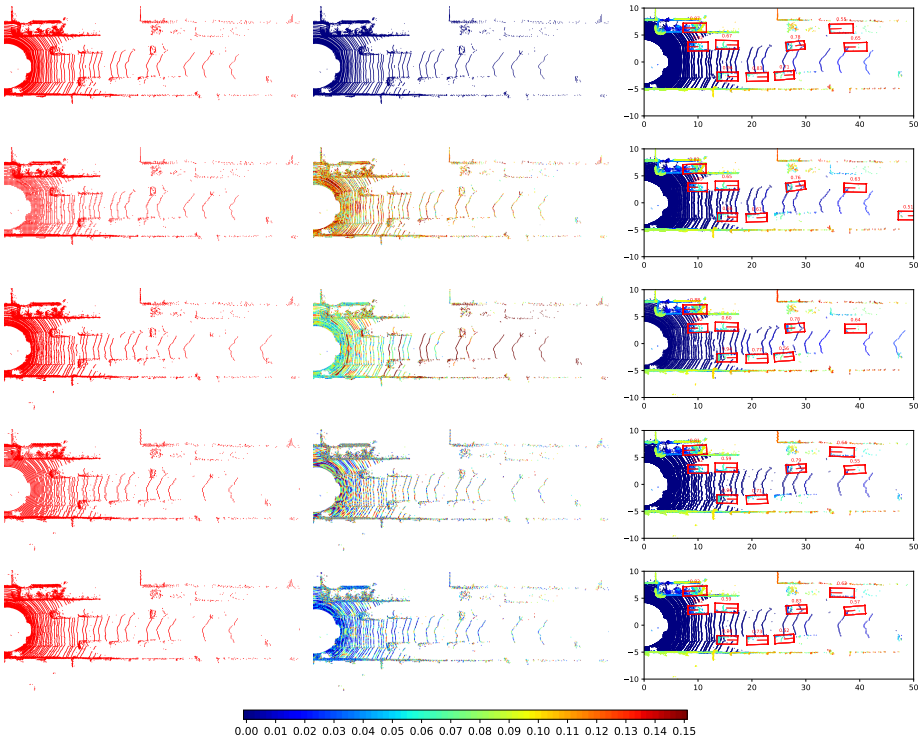

Fig. 3. The qualitative results of the reconstructed point cloud and the detection results of *KITTI/training/velodyne/000443.bin*. From left to right, there are three point clouds: reconstructed point cloud, error map with the error bar, and PointPillar detection results using OpenPCDet [6]. From top to bottom, there are the original point cloud with four comparative algorithms: Draco, G-PCC, R-PCC, and Ours. The BPP, compression ratio and chamfer distance results of each method are in the corresponding row of Tab. 3.

	BPP ↓	CR ↑	CD ↓
Original	96	1	0
Draco	1.63	58.83	0.064
G-PCC	2.2	43.60	0.065
R-PCC	1.71	56.06	0.046
Ours	**1.62**	**59.25**	**0.026**

Table 3. The BPP, compression rate, and mean chamfer distance (m) of each corresponding subfigure in Fig. 3 (corresponding rows).

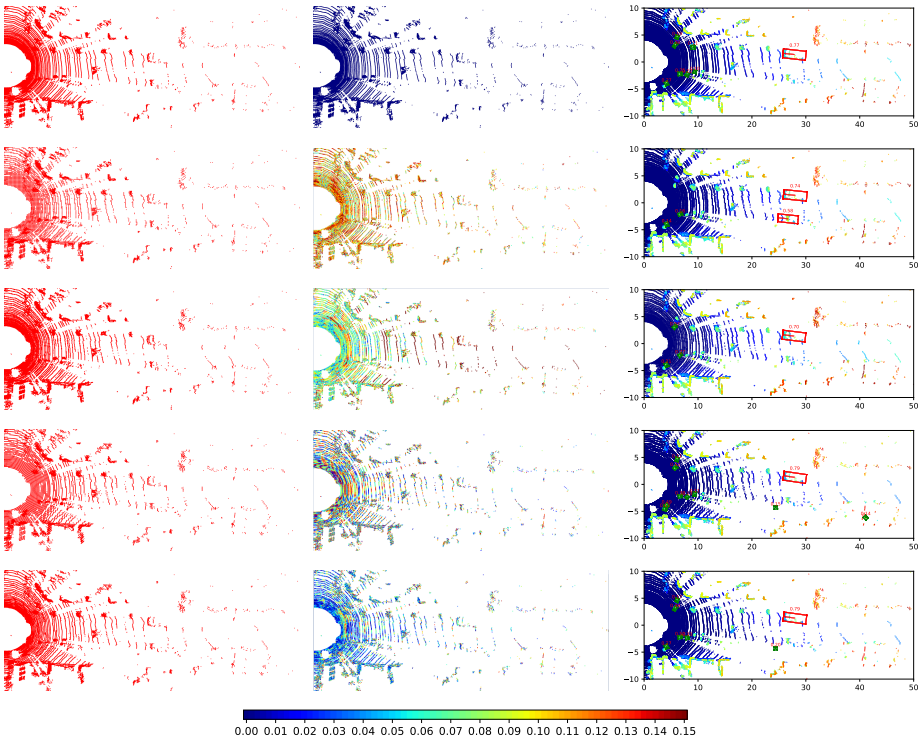

Fig. 4. The qualitative results of the reconstructed point cloud and the detection results of *KITTI/training/velodyne/000571.bin*. From left to right, there are three point clouds: reconstructed point cloud, error map with the error bar, and PointPillar detection results using OpenPCDet [6]. From top to bottom, there are the original point cloud with four comparative algorithms: Draco, G-PCC, R-PCC, and Ours. The BPP, compression ratio and chamfer distance results of each method are in the corresponding row of Tab. 4.

	BPP ↓	CR ↑	CD ↓
Original	96	1	0
Draco	**2.35**	**40.84**	0.063
G-PCC	2.99	32.07	0.068
R-PCC	2.67	35.94	0.046
Ours	2.47	38.80	**0.032**

Table 4. The BPP, compression rate, mean chamfer distance (m) of each corresponding subfigure in Fig. 4 (corresponding rows).

References

1. Choy, C., Gwak, J., Savarese, S.: 4d spatio-temporal convnets: Minkowski convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3075–3084 (2019)
2. Huang, L., Wang, S., Wong, K., Liu, J., Urtasun, R.: Octsqueeze: Octree-structured entropy model for lidar compression. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1313–1323 (2020)
3. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
4. Que, Z., Lu, G., Xu, D.: Voxelcontext-net: An octree based framework for point cloud compression. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6042–6051 (2021)
5. Smith, L.N., Topin, N.: Super-convergence: Very fast training of neural networks using large learning rates. In: Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications. vol. 11006, p. 1100612. International Society for Optics and Photonics (2019)
6. Team, O.D.: Openpcdet: An open-source toolbox for 3d object detection from point clouds. `https://github.com/open-mmlab/OpenPCDet` (2020)