Supplementary Material – OctSqueeze: Octree-Structured Entropy Model for LiDAR Compression

Lila Huang[1,2] Shenlong Wang[2,3] Kelvin Wong[2,3] Jerry Liu[2] Raquel Urtasun[2,3]
[1]University of Waterloo [2]Uber Advanced Technologies Group [3]University of Toronto
{lila.huang,shenlong.wang,kelvin.wong,jerryl,urtasun}@uber.com

Abstract

In this supplementary material, we describe additional experimental results that further validate the efficacy of our proposed method. We also benchmark the runtime of our proposed method and demonstrate its ability to encode LiDAR point clouds in real-time. Moreover, we exhibit an extensive array of qualitative results on NorthAmerica and KITTI that compares our method against Draco in terms of reconstruction quality and downstream task performance. Finally, we attach a video that presents an overview of our approach and showcases the quality of our point cloud reconstructions.

1. Additional Ablation Studies

We conduct a more thorough analysis on our entropy model to validate our choice of model architecture and the feature set we use. In Sec. 1.1, we show that our model performs best when using $K = 4$ levels of aggregation. Then, in Sec. 1.2, we demonstrate that our model's performance improvements arise as a result of our hierarchical feature aggregation scheme, and not because of the increase in model capacity. Finally, in Sec. 1.3, we present an expanded ablation study on our input feature set at the best level of aggregations; *i.e.*, $K = 4$. All experiments are conducted on the NorthAmerica evaluation set.

1.1. Number of Aggregations

# Aggregations	Bitrate		
	Depth = 12	Depth = 14	Depth = 16
0	3.48	8.91	14.97
1	3.39	8.78	14.84
2	3.31	8.59	14.64
3	3.25	8.47	14.51
4	**3.17**	**8.32**	**14.33**
5	3.27	8.51	14.55

Table 1: Ablation study on the number of aggregations.

Tab. 1 extends Tab. 2 in the main paper with an additional row entry for $K = 5$ aggregations. We found that $K = 5$ aggregations performs worse than $K = 4$ in terms of bitrate reduction, suggesting that our choice of $K = 4$ aggregations in the main paper is best for our architecture.

1.2. Aggregation of Parental Context Features

We also investigate whether a model that does not aggregate parental context features can achieve similar bitrate reductions as our model with $K = 4$ aggregations, holding all else equal. To perform this study, we trained an entropy model with the same architecture as our model with $K = 4$ aggregations, except that each node takes in a copy of its own context feature in the aggregation stage, rather than that of its parent. Tab. 2 shows our results. Surprisingly, we found that not only did

Parental Aggregations	K	Bitrate		
		Depth = 12	Depth = 14	Depth = 16
	0	3.48	8.91	14.97
	4	3.47	8.92	14.98
✓	4	**3.17**	**8.32**	**14.33**

Table 2: We compare the performance of our entropy model with and without aggregating parental context features (bottom two rows). Both models have the same model capacity as one with $K = 4$ aggregations. For completeness, we also show the performance of our model with $K = 0$ aggregations.

the model without parental aggregation perform worse than the one with aggregation, it also performed only as well as our original, smaller capacity model with $K = 0$ aggregations! This result suggests that adding more layers to the network alone does not translate to performance gains. Moreover, it validates our design of a tree-structured entropy model that progressively incorporates parental information through aggregations.

1.3. Input Context Features

L	P	O	LL	Bitrate		
				Depth = 12	Depth = 14	Depth = 16
✓				3.86	9.79	15.91
✓	✓			3.44	8.89	14.94
✓	✓	✓		3.34	8.72	14.76
✓	✓	✓	✓	**3.17**	**8.32**	**14.33**

Table 3: Ablation study on input context features for our model with $K = 4$ aggregations. L, P, O, and LL stand for the node's octree level, its parent occupancy symbol, its octant index, and its spatial location respectively.

We conduct an ablation study on the input context features used by our entropy model at $K = 4$ aggregations: the node's octree level, its parent occupancy symbol, its octant index, and its spatial location. Note that in Tab. 1 of the main paper, we presented an analogous ablation study on a model with $K = 0$ aggregations. In Tab. 3, we observe a similar decrease in bitrate as we increase the number of input context features. This result further corroborates our hypothesis that all four input context features contribute to the predictive power of our entropy model.

2. Additional Baselines

We conduct experiments comparing our compression method with two additional baselines. In Sec. 2.1, we experiment with a range view-based compression method that leverages the popular JPEG2000 image codec. Then in Sec. 2.2, we present results from our experiments with the voxel-based point cloud compression algorithm by Quach *et al*. [2].

2.1. JPEG Range Encoder

We compare against two baselines that use a range image representation of the input point cloud: Deep Range and JPEG Range. Deep Range is the range view-based method discussed in Sec. 4.2 of the main paper. In particular, given a LiDAR point cloud, we first construct a range image by converting it from Euclidean coordinates to polar coordinates, and then storing it as a 2.5D range image. Deep Range then uses a Ballé hyperprior model [1] to compress the 2.5D range image. In contrast, JPEG Range uses the popular JPEG2000 image codec to compress the 2.5D range image.

As shown in Fig. 1, Deep Range outperforms JPEG Range across all reconstruction quality metrics on both NorthAmerica and KITTI. This is a testament to the performance of deep learning-based methods for image compression. Moreover, as we alluded to in Sec. 4.4 of the main paper, our approach significantly outperforms both Deep Range and JPEG Range owing to its use of an octree data structure to represent the LiDAR point cloud and an octree-structured entropy model to compress it.

2.2. Deep Voxel Encoder

We additionally implemented the voxel-based point cloud compression algorithm by Quach *et al*. [2], consisting of a deep 3D convolutional autoencoder architecture with a fully-factorized entropy model inspired from [1]. We trained and

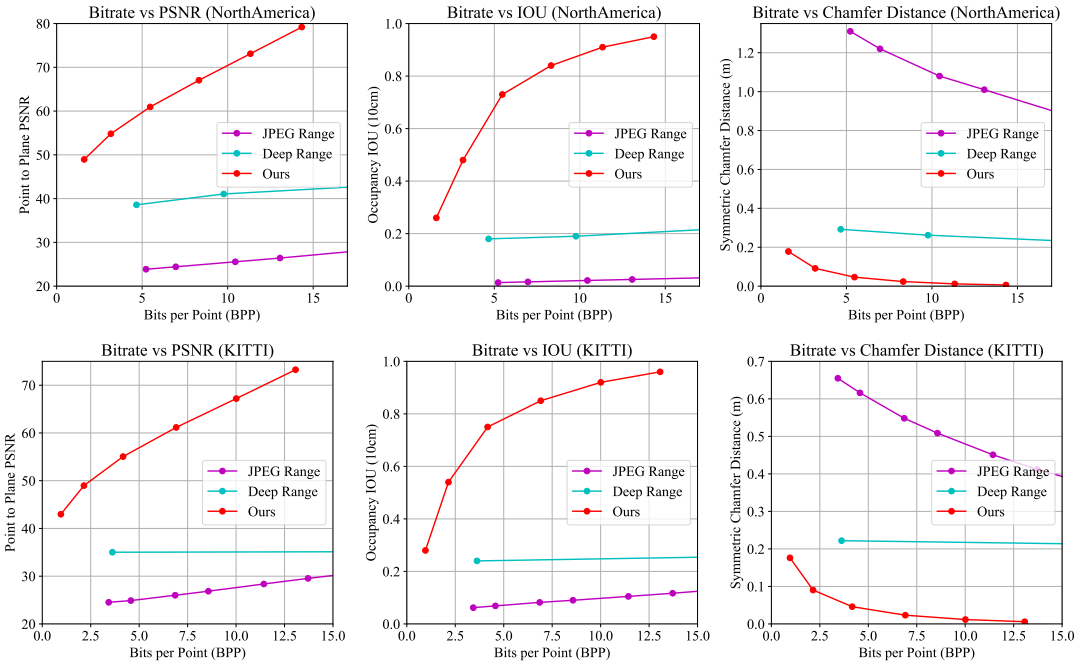

Figure 1: Quantitative results on NorthAmerica and KITTI. From left to right: point-to-plane PSNR, IOU, and Chamfer distance.

evaluated these models on the NorthAmerica LiDAR point cloud dataset, voxelizing points at (0.25m, 0.25m, 1.0m) for length, width, and depth dimensions respectively. Our best-performing reference model achieves a point-to-plane PSNR of 33.76 at a bitrate of 26.81—this performs much worse than the tree-based methods of Draco (PSNR: 48.47, bpp: 2.778) and our approach (PSNR: 48.95, bpp 1.61). The underperformance of the voxel-based compression method indicates that a dense voxel representation may not be the best fit for compressing LiDAR point clouds due to the inherent sparsity and high frequency information in this data.

3. Runtime

Depth	Encoding (*ms*)				Decoding (*ms*)
	Octree	Network	Range Coding	Total	Total
10	14.49	7.47	0.62	22.58	53.31
11	21.10	13.85	0.80	35.75	59.53
12	23.73	24.67	1.18	49.58	95.01
13	25.51	39.82	2.19	67.52	138.81
14	32.34	56.04	3.15	91.53	140.78
15	34.85	65.17	3.55	103.57	147.89
16	35.52	66.83	3.61	105.96	150.74

Table 4: Runtime of our model with $K = 4$ aggregations (in *milliseconds*). 'Depth' is the maximum depth of the octree. 'Octree' is the time to build the octree; 'Network' the time to run our entropy model; and 'Range Coding' the time of range coding.

We benchmarked our approach on a workstation with an Intel Xeon E5-2687W CPU and a Nvidia GeForce GTX 1080 GPU. See Tab. 4 for the results. In our experiments, octree building and range coding were implemented in C++, and our entropy model was implemented in Python with PyTorch. Our approach achieves end-to-end encoding in real-time, meaning that our algorithm can be deployed in an online setting. Moreover, our decoding speeds are quite fast as well; due to the dependence of each node on ancestral nodes, we interleave range decoding and octree construction with GPU model forward passes for each level.

4. Additional Qualitative Results

We exhibit an extensive array of qualitative results that compare our method against Draco across a spectrum of bitrates. In Fig. 2 and 3, we show the reconstruction quality of our method versus Draco. Then, in Fig. 4 and 5, we show their respective downstream semantic segmentation performance. Finally, in Fig. 6, we show their respective downstream object detection performance. As indicated in these figures, our model can attain better results than Draco at comparable—or even lower—bitrates.

We also showcase our method in the form of a video. In this video, we highlight a visual sequence of LiDAR point clouds captured by a self-driving vehicle in the NorthAmerica dataset. We compare the *Oracle* (*i.e.*, uncompressed point clouds) and our reconstructions side-by-side, and note that our approach achieves high visual similarity to the original point cloud at a much lower bitrate. Additionally, we also showcase semantic segmentation and object detection results produced using the Oracle and our reconstructed point clouds. The outputs produced using our reconstructed point clouds are almost identical to those produced using the Oracle, again validating the practicality of our compression method.

References

[1] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. 2

[2] Maurice Quach, Giuseppe Valenzise, and Frédéric Dufaux. Learning convolutional transforms for lossy point cloud geometry compression. In *2019 IEEE International Conference on Image Processing, ICIP 2019, Taipei, Taiwan, September 22-25, 2019*, pages 4320–4324. IEEE, 2019. 2

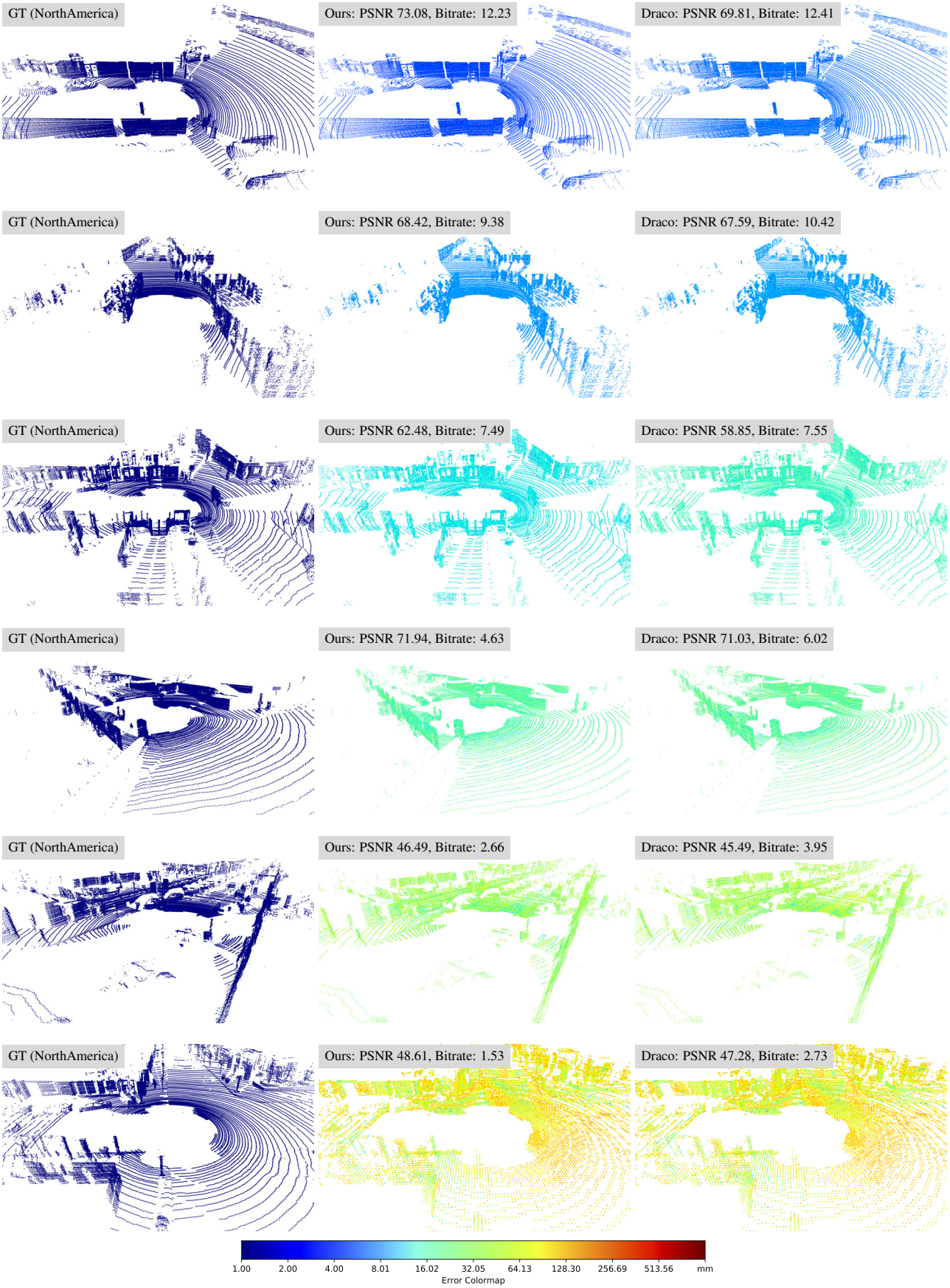

Figure 2: Qualitative results of reconstruction quality for NorthAmerica.

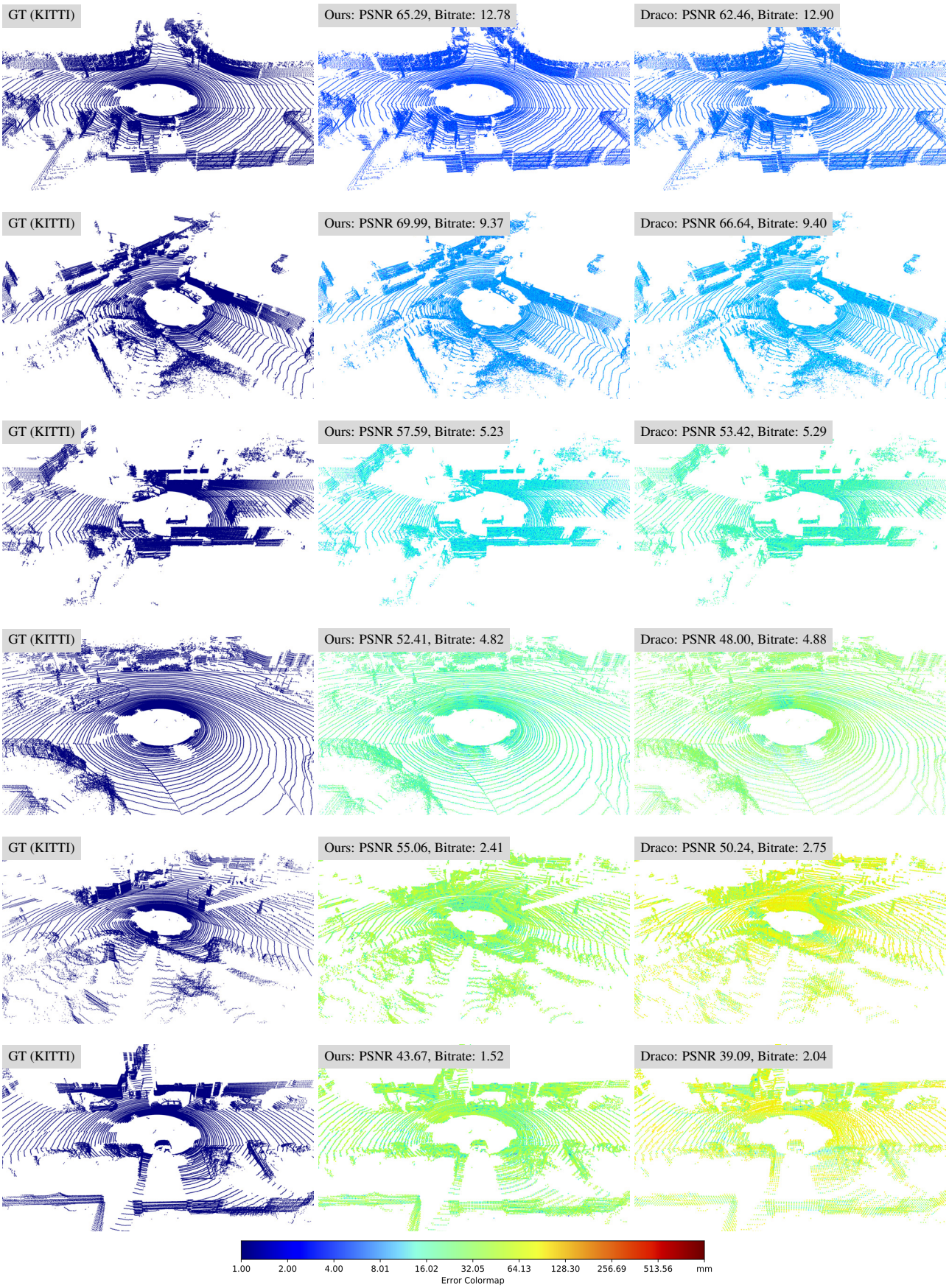

Figure 3: Qualitative results of reconstruction quality for KITTI.

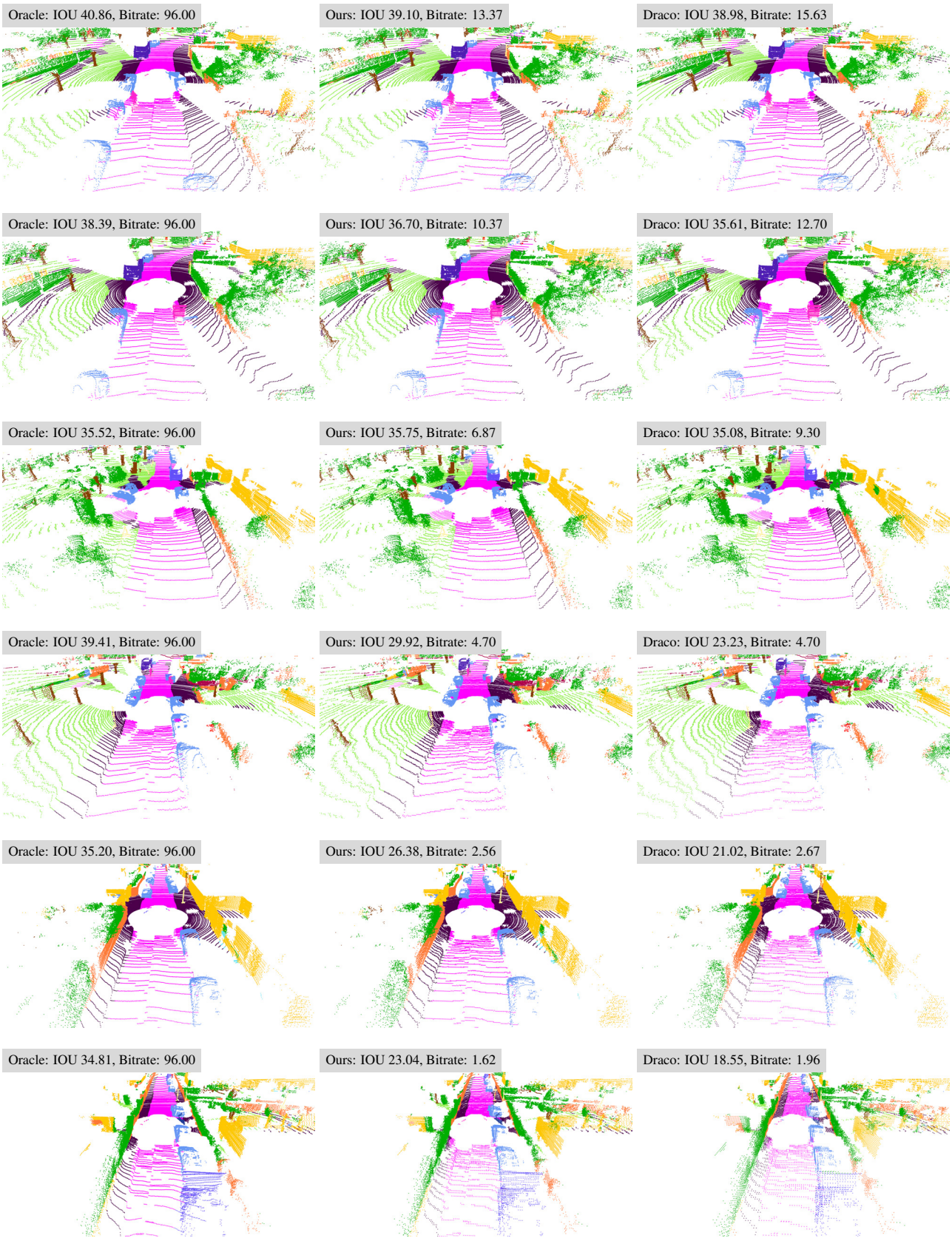

Figure 4: Qualitative results of semantic segmentation for KITTI. IOU is averaged over all classes.

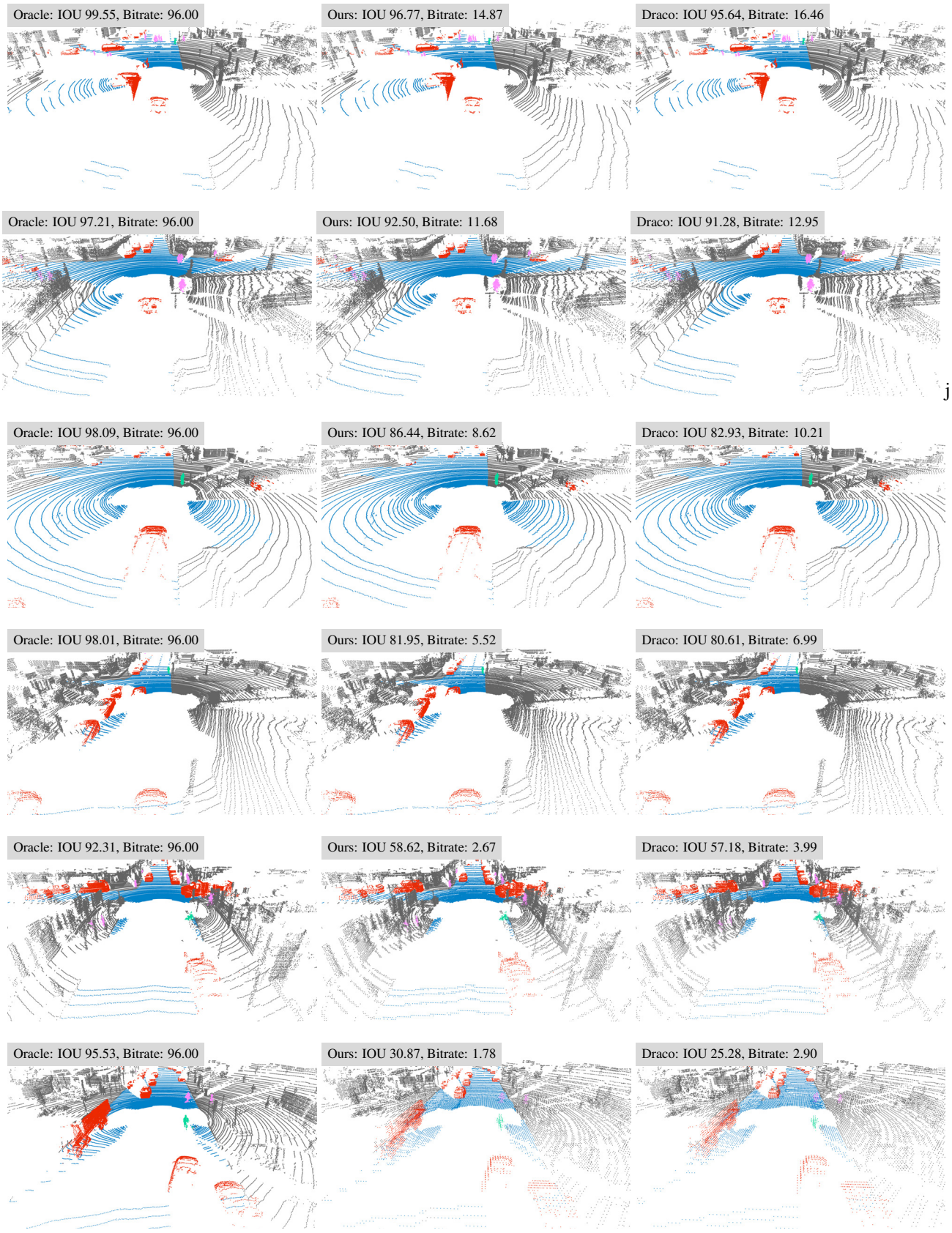

Figure 5: Qualitative results of semantic segmentation for NorthAmerica. IOU is averaged over all classes.

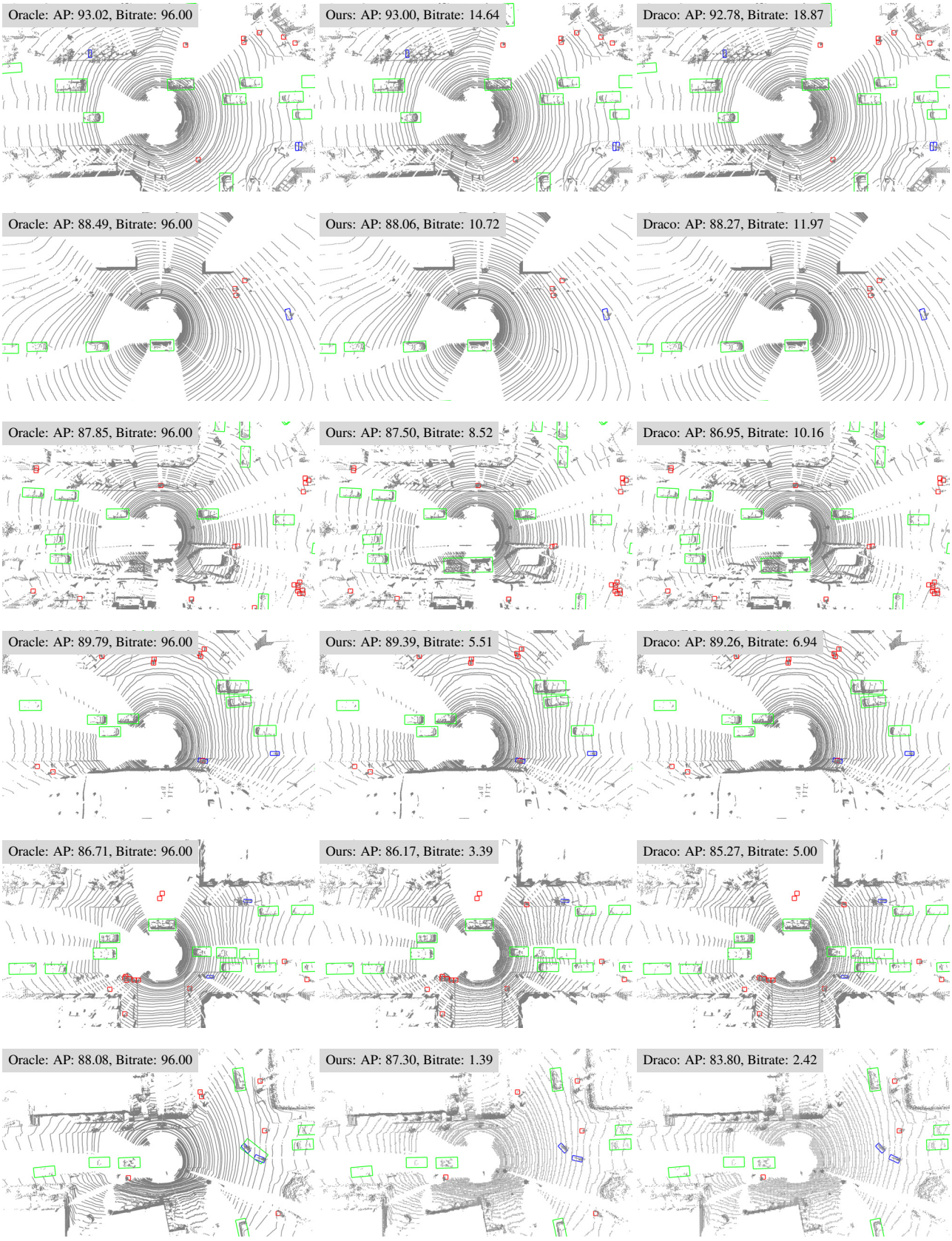

Figure 6: Qualitative results of object detection for NorthAmerica. AP is averaged over vehicle, motorbike, and pedestrian classes.