We thank the reviewers for their valuable comments. In the following, we will address each comment in detail.

R1, R2: Novelty. To our knowledge, learning-based density models of point cloud sweeps [spatial+intensity] for compression are not widely explored, and modeling temporal relationships between point clouds is non-trivial. Our novel tree-structured density models combine existing techniques (deep sets, cont. convs) to significantly advance beyond the scope of OctSqueeze – fully exploiting temporal redundancies and modeling intensity attributes.

R2: Additional baselines. We benchmarked MPEG TMC13 on UrbanCity and SemanticKITTI (see Fig. 1). On average, our approach achieves a better bitrate vs. reconstruction trade-off, especially at lower bitrates. We could not benchmark MPEG TMC2 due to `segmentation fault` errors in their reference implementation [1]. We have emailed the authors and hope to include TMC2 in our final paper. That said, TMC2 is related to our MPEG Range baseline: both compress 2D projections of the point cloud using video codecs. We found that tree-based codecs perform better.

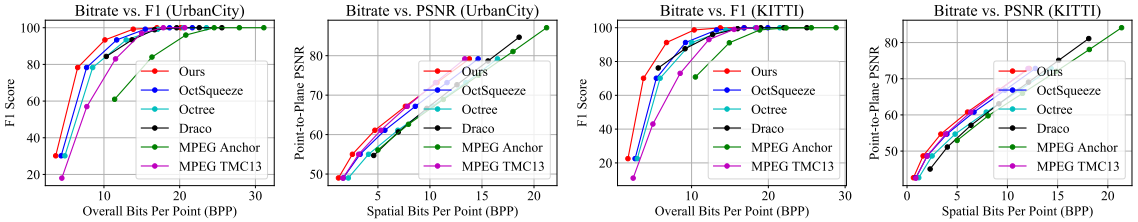

Figure 1: Bitrate vs. reconstruction quality on UrbanCity and SemanticKITTI.

R2, R3: Runtime/GPU passes. We benchmarked our approach on a workstation with an Nvidia GTX 1080Ti GPU and Intel Xeon E5 CPU. Our octree data structure and range coder is written in C++ for CPU execution, and our spatial/intensity entropy models run on the GPU. During encoding, we predict distributions over occupancy symbols with one forward pass of the entropy models. During decoding, we require $O(D)$ forward passes (D is the octree's depth), each interleaved with range decoding to reconstruct the octree level by level. We also benchmarked OctSqueeze, Draco, MPEG Anchor, and MPEG TMC13. Our encoding speed is fast, and our total runtime is on par with TMC13. Moreover, there are many opportunities to speed up our research code significantly, especially in CPU/GPU I/O.

	MuSCLE (Spatial)		MuSCLE (Intensity)		OctSqueeze (Spatial)		Draco (Spatial)		MPEG Achor (Spatial + Intensity)		MPEG TMC13 (Spatial + Intensity)	
Depth	Encoding (*ms*)	Decoding (*ms*)	Encoding (*ms*)	Decoding (*ms*)	Encoding (*ms*)	Decoding (*ms*)	Encoding (*ms*)	Decoding (*ms*)	Encoding (*ms*)	Decoding (*ms*)	Encoding (*ms*)	Decoding (*ms*)
11	65.6	279.5	20.8	49.0	16.5	93.0	25.8	15.3	59.6	33.6	1219.0	272.0
12	94.4	477.5	31.4	89.9	26.2	165.7	29.0	16.2	84.2	50.3	1906.9	461.4
13	135.6	764.0	42.7	137.9	40.1	299.4	30.3	17.2	106.5	67.2	2539.0	674.4
14	177.1	1084.2	49.2	162.6	53.4	486.4	30.7	17.5	125.5	84.1	3038.9	839.1
15	202.4	1352.1	50.7	166.3	63.6	699.0	30.9	17.6	143.0	98.9	3406.3	925.8
16	208.4	1569.4	55.5	165.6	65.4	902.3	31.1	17.6	160.2	112.8	3695.1	985.0

Table 1: Encoding/decoding time (*ms*) of MuSCLE spatial intensity model, OctSqueeze, Draco, MPEG Anchor, and MPEG TMC13.

R2, R3: Experiment details. The average number of points per LiDAR point cloud is 80,156 in UrbanCity and 120,402 in SemanticKITTI. During training, our full model requires 6-10GB of GPU memory. 16 GPUs was our starting point, but we retrained models with 4 GPUs and found roughly equal convergence / test metrics.

R1: Clarification of line 138 and bitrates. The $\sigma\colon \mathbb{R}^3 \to \mathbb{R}^d$ in line 138 is an MLP that predicts d-dimensional weights for continuous convolution. Bitrate is the total number of bits used to store the LiDAR dataset divided by the total number of points it contains. We do not count the one-time transmission of network weights (standard practice in learned compression) since this is negligible compared to the size of long LiDAR streams, e.g., 1 hour. We emphasize that the middle and right columns of Fig. 2 depict spatial-only bitrates since they compare spatial reconstruction quality.

R4: Typos and clarification of line 71. Thanks, we will fix the typos. Regarding line 71, our quantization scheme may produce a point cloud with fewer points than the original point cloud. Thus, we determine the intensity value of each point in the quantized point cloud by taking that of its nearest neighbour in the original point cloud.

R4: Simpler entropy models. Our tree-based baselines use a spectrum of statistical models with varying complexity. For example, Octree and MPEG Anchor use an empirical histogram-based entropy model, MPEG TMC13 uses a cascade of adaptive frequency tables, and OctSqueeze uses a deep learning model.

R4: Invariance to scale, density, and tree depth. Thanks for the exciting insight! We agree that ideally our density estimation model based on tree-structured message passing and continuous convolutions should be invariant to scale, density, and tree depth, echoing R4's thoughts on potential applications to other perception tasks. However, in our current implementation, several mechanisms make this not the case: 1) our input feature include information on the depth; 2) we use separate continuous convolution modules per depth; and 3) we do not use data augmentation (e.g., random scaling) and so our network learns the inductive bias of the dataset (e.g., the real-world scale). That said, by modifying these mechanisms, we should get a density model that is invariant to scale, density, and tree depth.

[1] MPEG TMC13: `github.com/MPEGGroup/mpeg-pcc-tmc13`, MPEG TMC2: `github.com/MPEGGroup/mpeg-pcc-tmc2`.