On the Feasibility and Opportunity of Autoregressive 3D Object Detection

Supplementary Material

In this supplementary material, we provide more details and experiment results in addition to the main paper:

- Sec. A: Future works and discussions.
- Sec. B: Details on Cascading Refinement methodology.
- Sec. C: Additional experiments.
- Sec. D: Additional details on training and inference.
- Sec. E: Additional quantitative results.
- Sec. F: Additional qualitative results.

A. Future Works and Discussions

A.1. Future Directions

Our work focuses on demonstrating the viability of autoregressive 3D point-cloud detection. Further experiments should explore model scaling to larger datasets and architectures [5, 18]. Additionally, while we leverage a semantically expanded vocabulary, learned codebooks [62] that are specific to object detection could improve token efficiency and representation flexibility. These directions were beyond our computational budget but represent natural extensions.

A.2. Inference Speed and Applicability

A shared limitation of the autoregressive formulation is inference latency. Despite using bfloat-16 precision and KV-caching, our implementation achieves $\sim$5 Hz under batched inference, corresponding to 1–2 Hz for single-scene inference with voxel-based backbones. While optimization techniques from language modeling exist, their effectiveness for detection remains unexplored, particularly for the LiDAR domain.

A.3. Opportunities via Autoregressive Detectors

By formulating object detection as autoregressive modeling, we connect 3D detection to broader sequence modeling advances. Test-time scaling techniques [16, 19, 44] that trade inference compute for performance may apply to detection. Furthermore, autoregressive representations could facilitate integration with language [18, 50, 60] and vision-language models [5, 35, 38]. This alignment could enable spatial-linguistic reasoning and allow LLMs to leverage 3D spatial data in pretraining. However, achieving such cross-modal alignment requires addressing fundamental differences in data and task objectives not explored in this work. We view our contribution as establishing autoregressive detection viability as a foundation for these future research directions.

A.4. Discussion on 3D Detection Heads

Historically, 3D object detection heads have evolved through three main paradigms to address the challenges of accurate bounding box prediction. Anchor-based detection heads [34, 58, 71, 79] extend 2D detection principles by utilizing predefined 3D bounding box templates distributed across feature space, achieving strong performance but suffering from hyperparameter sensitivity regarding anchor configurations and requiring careful tuning for different object categories. To address these limitations, center-based detection heads emerged, directly predicting object centers and associated properties. CenterPoint [76] pioneered this by detecting object centers as keypoints in bird's-eye view before regressing to 3D boxes, while FCOS3D [68] extended this to monocular detection. AFDetV2 [28] and FCAF3D [55] demonstrated superior LiDAR-based performance without anchor matching overhead. While these methods eliminate hand-crafted anchor designs and handle varying object sizes naturally, they still require post-processing like NMS. More recently, DETR-style heads [11] leverage transformers with learnable object queries. DETR3D [69] introduced sparse queries that predict 3D boxes through cross-attention with multi-view features, while PETR [40] added position-embedded queries and BEVFormer [36, 74] combined this with bird's-eye view representations. These query-based methods eliminate anchors and NMS, enabling end-to-end training and modeling long-range dependencies. However, they typically require complex query initializations and suffer from limited architectural flexibility due to reliance on fixed query-target matching mechanisms. In contrast, AutoReg3D offers a flexible autoregressive formulation, removing the complex target assignment, anchors, and post-processing thresholds, resulting in a simplified training and inference pipeline.

A.5. Discussion on Model Architecture

In this work, we present a simple and straightforward encoder–decoder design for autoregressive 3D detection. A natural alternative is to directly feed point-cloud features into a decoder-only Transformer via self-attention [38]. We conducted preliminary experiments with a decoder-only formulation using a Q-Former [35] to compress point-cloud features. However, this approach did not yield meaningful performance improvements over the simpler encoder–decoder formulation and additionally exhibited reduced training stability.

Since our goal is to demonstrate a simple and clean solution to 3D autoregressive detection, we focus on the en-

coder–decoder design. We leave the exploration of more complex or optimized architectural variants to future work.

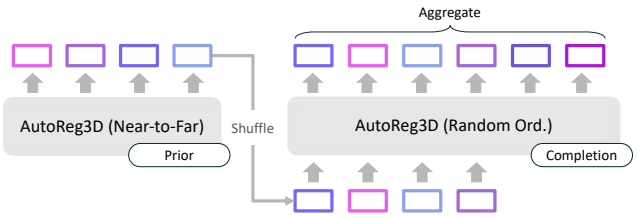

Figure A1. **Cascading Refinement Methodology.**

B. Cascading Refinement

We illustrate the methodology for ***Cascading Refinement*** from Section 3.2 of the main text in Figure A1. For our setup, we take the output of the distance-ordering AutoReg3D ("prior") model and use those as context for a random-ordering AutoReg3D ("completion") model after shuffling to eliminate the near-to-far positional bias. The completion model generates additional boxes conditioned on the given context. The generations are then aggregated using a simple IoU-based clustering strategy. Specifically, we group boxes that have IoU over a selected threshold, and we average the box parameters within each group. We set the IoU threshold to be 0.1. Results are reported in Table 6 of the main text. Additional qualitative results for Cascading Refinement are provided in Figure A3 and Figure A4.

C. Additional Analysis

C.1. Ordering Comparison by Distance

We analyze the performance differences between the random-order model and the near-to-far model. As shown in Table A1, the near-to-far model outperforms the random-order model across all distance bins and all metrics. Moreover, the F1 gap between the two models grows with distance from the ego vehicle. By generating the nearby objects first, the near-to-far model is able to better reason about farther objects using previously predicted boxes as hints. In contrast, the random-order model does not leverage this structure, limiting its ability to detect distant objects, leading to a widening performance gap at larger distances. These results highlight the importance of modeling interdependencies between objects in autoregressive 3D detection.

C.2. Tokenization Methods

We analyze different tokenization strategies for discretizing box parameters.

Shared *vs*. Expanded Vocabulary. We compare a shared vocabulary across all attributes and an expanded vocabulary that assigns separate token ranges to each parameter type. In the shared-vocabulary setting, all continuous parameters are uniformly quantized into an integer between 1 and 2160. As shown in Table A2, both strategies achieve similar performance, indicating that our method is robust to the choice of vocabulary. We adopt the expanded vocabulary because it aligns more naturally with the heterogeneous semantics and numeric ranges of different box attributes (see Figure 3 of the main text). Separating the token spaces makes token-type masking and decoding logic more explicit and reduces ambiguity about which parameter a given token represents.

	0–10m	10–20m	20–30m	30m+
Precision				
Random	77.5	72.8	64.1	49.2
Near-to-far	80.7	80.1	70.8	50.5
Recall				
Random	73.5	62.4	46.5	25.7
Near-to-far	81.6	72.5	55.6	32.1
F1				
Random	74.1	66.1	52.0	31.2
Near-to-far	79.6	75.8	61.7	38.2
Δ (%)	+7.5%	+14.6%	+18.8%	+22.7%

Table A1. **Ordering Comparison by Distance to Ego**. Near-to-far order outperforms random order across all metrics and distance bins.

Method	Precision	Recall	F1
Shared	74.2	59.7	65.7
Expanded	74.9	59.4	65.8

Table A2. **Expanded *vs*. Shared Vocabulary.**

Binning Strategies. To build our customized vocabulary, we uniformly quantize each continuous box parameter $x, y, z, l, w, h, \psi, v_x, v_y$, into 2160, 2160, 160, 600, 200, 200, 125, 600, 600 bins respectively, which is equivalent to quantize parameters with bin widths $0.05\,\mathrm{m}$ for center/size; $0.05\,\mathrm{rad}$ for yaw; and $0.1\,\mathrm{m/s}$ for velocity. A distribution over the token frequency can be seen in Figure A2.

To justify our choice of token resolution and bin count, we ablate different settings in Table A4. Overall, our method is largely robust to the choice of binning strategy. Reducing the bin count by half (coarser resolution) slightly improves F1, but worsens true-positive (TP) errors; we attribute this to the fact that nuScenes matching is center-distance based and therefore tolerant to small quantization errors, while a smaller vocabulary simplifies prediction, leading to the slight F1 gain. Doubling the bin count (finer resolution) yields lower F1, lower TP er-

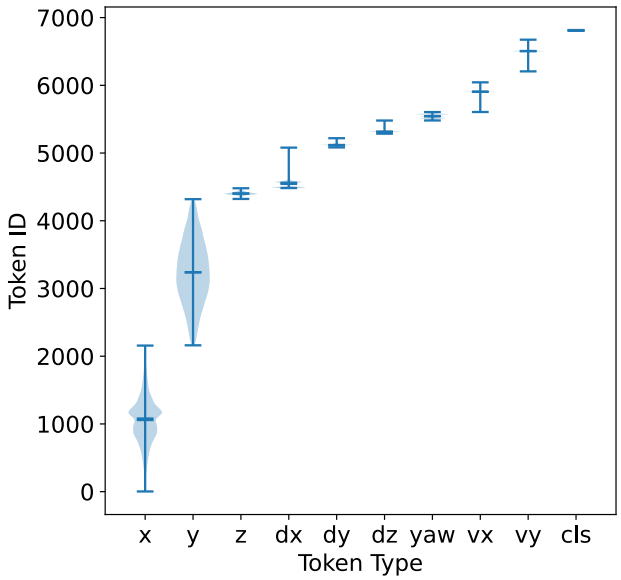

Figure A2. **Token Distribution After Quantization.** Violin plot of the token distribution, plotting token ID by type. Semantic types occupy disjoint ID ranges, with types that cover larger value ranges correspondingly taking up larger token ranges.

Method	Precision	Recall	F1
Nucleus	72.8	60.4	65.8
Greedy	74.5	60.9	66.7
Beam Search	74.4	60.9	66.7

Table A3. **Analysis on RL Decoding Method.** Different generation decoding strategies have limited effect on performance after RL fine-tuning.

rors, but higher memory usage. We also attempted adaptive (quantile-based) binning with the same number of bins, which achieves similar F1 to the default setting but worse TP errors, likely due to train-test distribution mismatch. Thus, our default choice strikes a balance between efficiency and performance.

C.3. RL Decoding Methods

We compare the effect of different decoding strategies on our method after RL fine-tuning in Table A3. Similar to teacher-forcing in Table 5, nucleus sampling (F1=65.8) performs worse than greedy (F1=66.7) and beam search (F1=66.7). However, unlike teacher-forcing, greedy decoding matches beam search due to RL's sharpening effect on token distributions, causing beams to collapse to similar trajectories, as reported by others [77]. This sharpening enables the RL model to outperform teacher-forcing (F1=61.9) under nucleus sampling.

D. Additional Details on AutoReg3D

D.1. Additional Implementation Details

For supervised training, we train all models for 20 epochs with a batch size of 64. We first freeze the encoder and only train the decoder, then we unfreeze and train both the encoder and decoder jointly. We apply a cosine warm-up schedule for the first 10% of the total training steps. For RL fine-tuning, we train the decoder with a fixed learning rate of 1×10^{-4} and no learning rate scheduling, while keeping the encoder frozen. We use a group size of 8, omit the KL penalty by setting $\beta = 0$, and train with a batch size of 64 for around 8 hours. We train all models using automatic mixed precision with BF16. To further speed up inference, we adopt KV-caching and an early stopping condition during decoding. Our implementation is based on OpenPCDet[1] and HuggingFace Transformers[2]. All experiments are conducted using 4 NVIDIA A100 (80GB) or 4 H100 GPUs.

E. Additional Quantitative Results

We show additional nuScenes quantitative results in Table A5, listing the threshold used for each baseline and TP errors of each method.

F. Additional Qualitative Results

We include additional qualitative results:

- Figure A3 & Figure A4: Qualitative results on Cascading Refinement.
- Figure A5: Qualitative results on pillar-based backbone.
- Figure A6: Qualitative results on voxel-based backbone.
- Figure A7: Qualitative results on transformer-based backbone.
- Figure A8: Qualitative results on Mamba-based backbone.
- Figure A9: Qualitative results on random-order model.
- Figure A10: Qualitative examples of AutoReg3D failure modes.

[1] https://github.com/open-mmlab/OpenPCDet
[2] https://github.com/huggingface/transformers

Method	Prec.	Rec.	F1	mATE ↓	mASE ↓	mAOE ↓	mAVE ↓	mAAE ↓
Ours	74.9	59.4	65.8	0.323	0.278	0.362	0.234	0.191
Ours ($0.5\times$bin count)	74.9	61.0	66.8	0.324	0.283	0.339	0.272	0.183
Ours ($2\times$bin count)	74.3	59.9	65.8	0.316	0.276	0.357	0.218	0.182
Ours (w/ adaptive binning)	73.7	59.9	65.7	0.323	0.299	0.363	0.257	0.202

Table A4. **Binning Strategy Ablation.**

Method	Encoder	Thresh.	Prec.	Rec.	F1	mATE ↓	mASE ↓	mAOE ↓	mAVE ↓	mAAE ↓
PointPillars	Pillar Conv.	-	3.4	85.8	6.2	0.353	0.263	0.346	0.290	0.198
CenterPoint		-	5.6	82.4	10.1	0.316	0.262	0.437	0.241	0.190
PointPillars		0.29	58.3	50.0	53.1	0.312	0.252	0.263	0.282	0.212
CenterPoint		0.50	67.9	53.3	59.5	0.277	0.251	0.381	0.229	0.203
Ours		-	69.6	52.4	59.2	0.365	0.285	0.424	0.262	0.195
SECOND	Voxel Conv.	-	3.8	85.9	6.9	0.325	0.259	0.286	0.268	0.198
CenterPoint		-	6.8	85.3	12.1	0.292	0.255	0.381	0.217	0.181
SECOND		0.29	63.5	55.6	59.1	0.289	0.248	0.219	0.254	0.214
CenterPoint		0.49	72.8	60.3	65.8	0.263	0.247	0.335	0.207	0.190
Ours		-	74.9	59.4	65.8	0.323	0.278	0.362	0.234	0.191
DSVT	Transformer	-	17.6	83.3	27.4	0.270	0.248	0.276	0.228	0.189
DSVT		0.27	79.1	66.3	71.6	0.253	0.242	0.243	0.223	0.198
Ours		-	77.0	64.1	69.5	0.295	0.281	0.304	0.238	0.194
LION	Mamba	-	16.7	84.9	26.3	0.265	0.244	0.271	0.231	0.187
LION		0.27	78.6	68.3	72.5	0.247	0.239	0.234	0.228	0.193
Ours		-	77.5	65.2	70.4	0.301	0.278	0.283	0.220	0.181

Table A5. **Detailed Evaluation Result.**

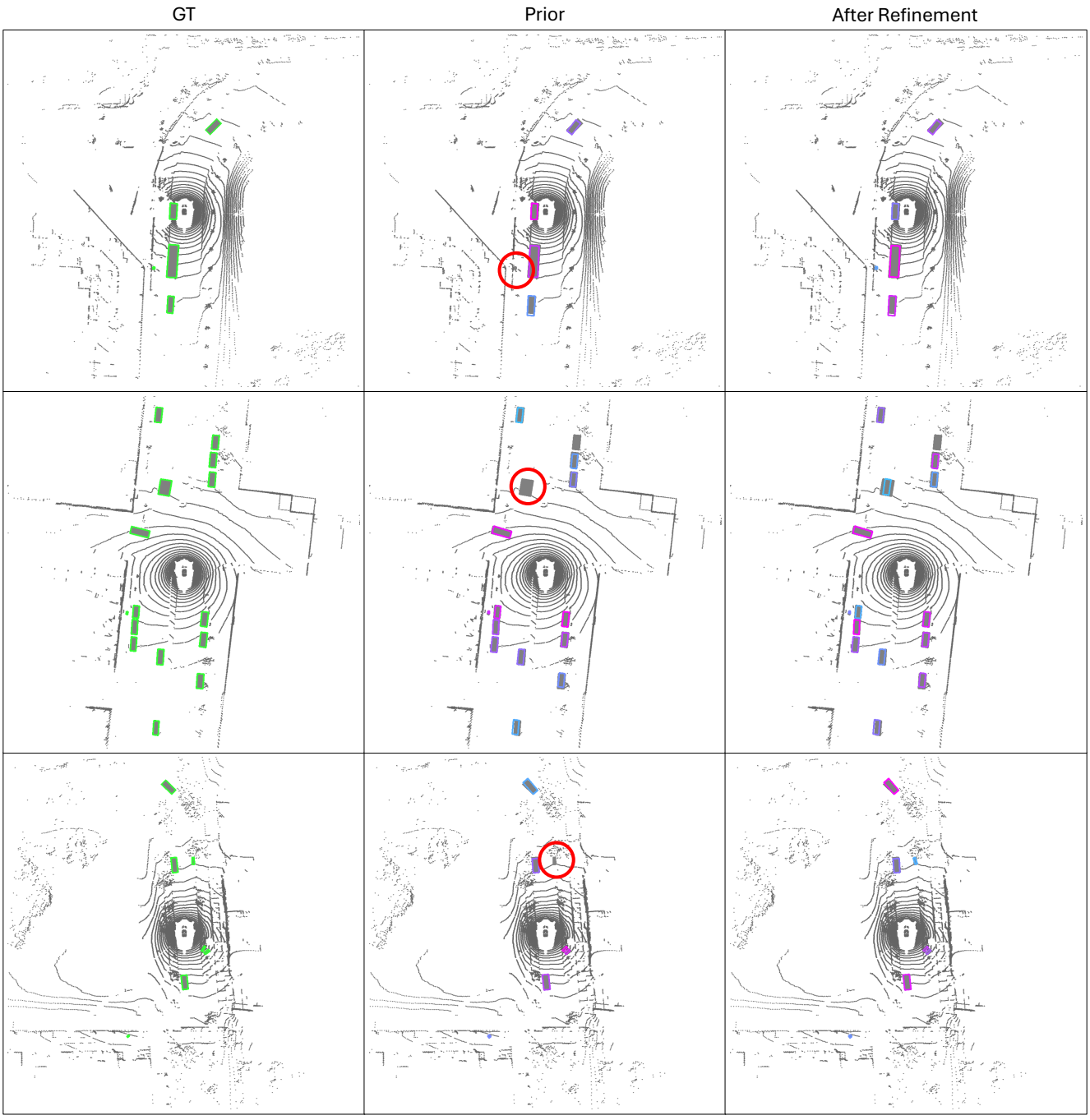

Figure A3. **Qualitative Results on Cascading Refinement.** We visualize the ground-truth boxes (left), predictions from the near-to-far prior model (middle), and predictions after Cascading Refinement (right). Cascading Refinement recovers objects that were missed by the prior model (circled in red). Predicted boxes are colored by generation order from first (magenta) to last (blue). Best viewed in color.

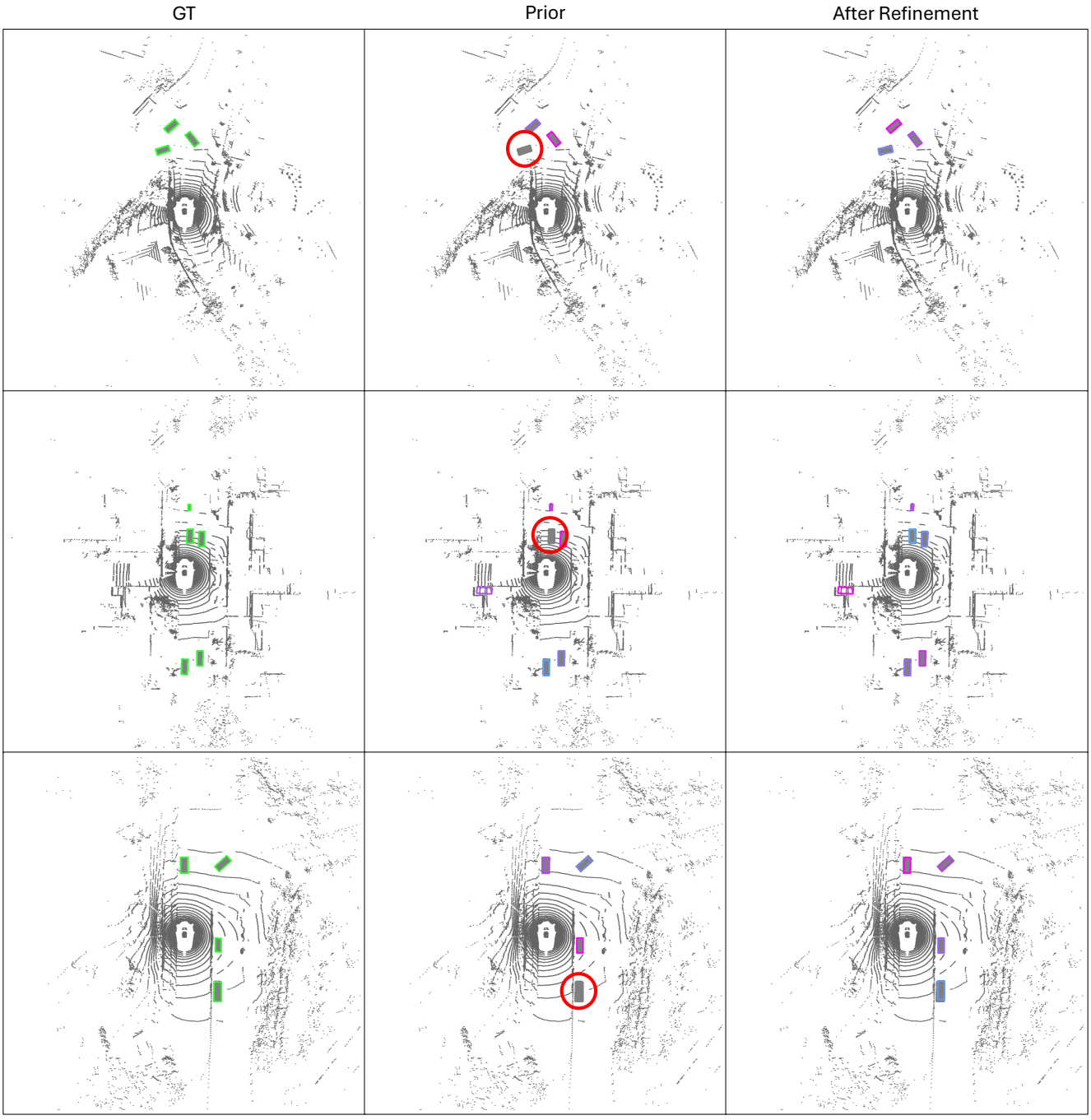

Figure A4. **Qualitative Results on Cascading Refinement.** We visualize the ground-truth boxes (left), predictions from the near-to-far prior model (middle), and predictions after Cascading Refinement (right). Cascading Refinement recovers objects that were missed by the prior model (circled in red). Predicted boxes are colored by generation order from first (magenta) to last (blue). Best viewed in color.

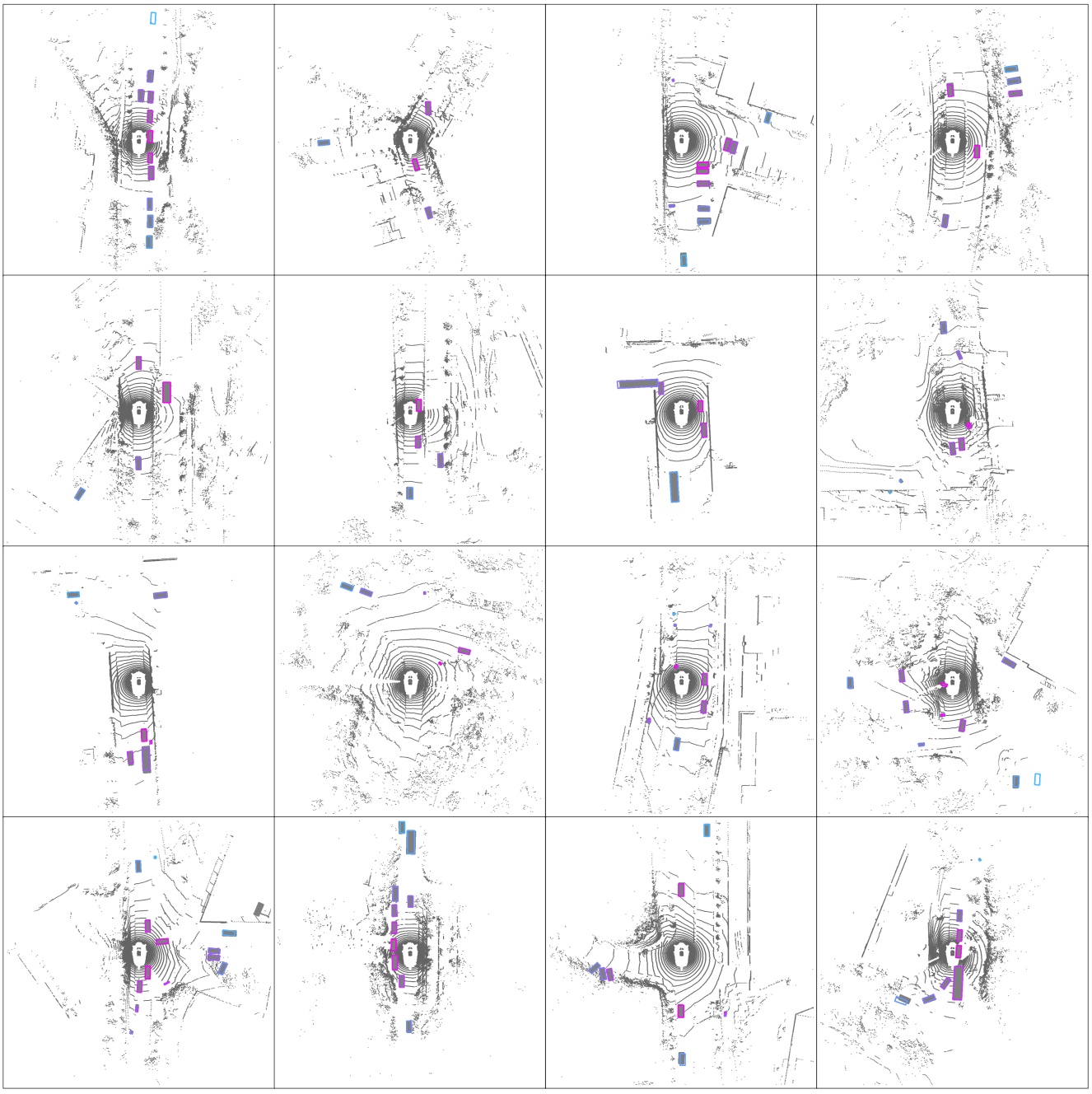

Figure A5. **Qualitative Results on Pillar-based Backbone.** We visualize bounding box generations from our method with pillar-based backbone trained to generate boxes in near-to-far order. AutoReg3D generates boxes from first (magenta) to last (blue), ground-truth boxes are in gray. Best viewed in color.

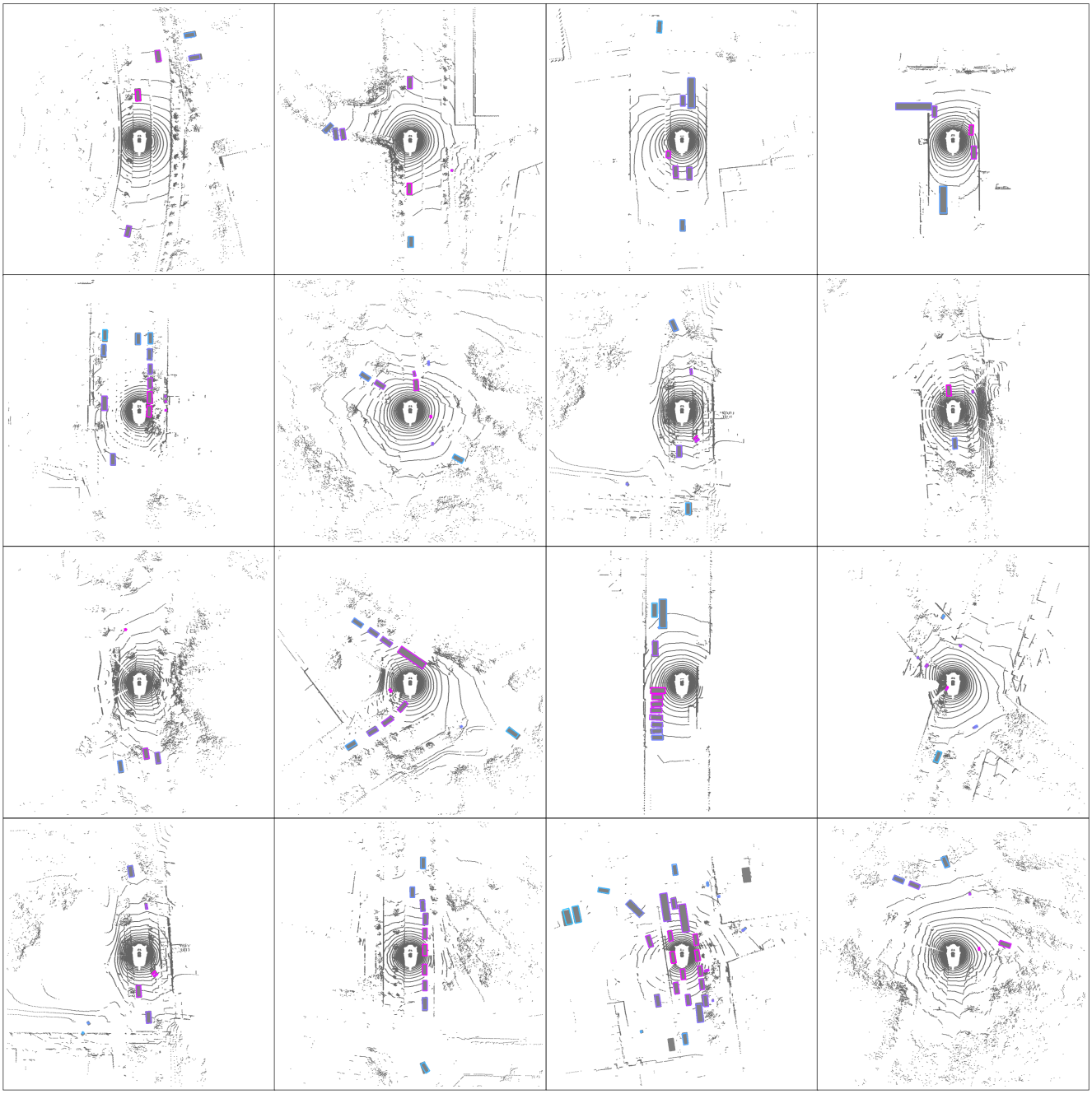

Figure A6. **Qualitative Results on Voxel-based Backbone.** We visualize bounding box generations from our method with voxel-based backbone trained to generate boxes in near-to-far order. AutoReg3D generates boxes from first (magenta) to last (blue), ground-truth boxes are in gray. Best viewed in color.

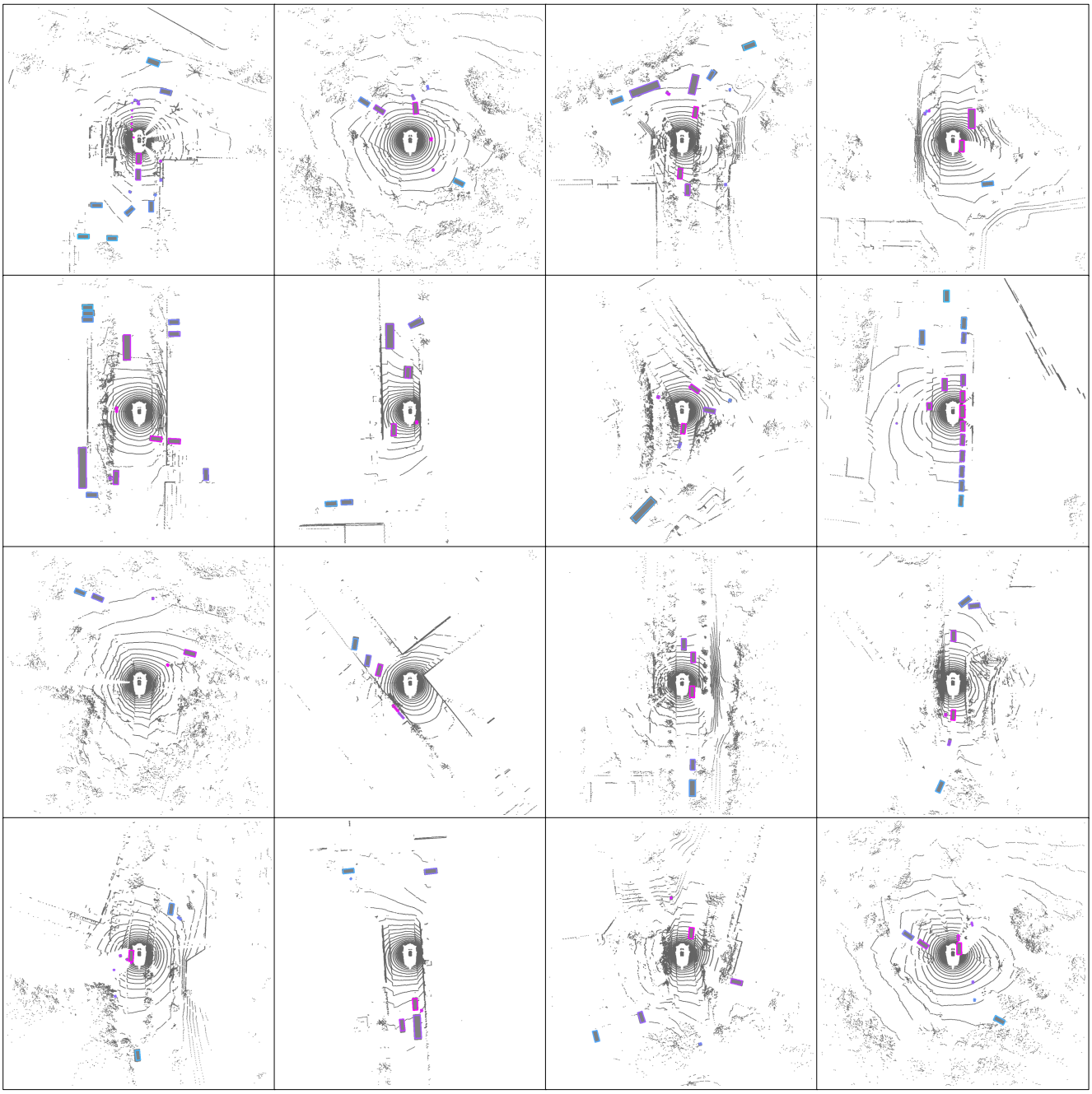

Figure A7. **Qualitative Results on Transformer-based Backbone.** We visualize bounding box generations from our method with transformer-based backbone trained to generate boxes in near-to-far order. AutoReg3D generates boxes from first (magenta) to last (blue), ground-truth boxes are in gray. Best viewed in color.

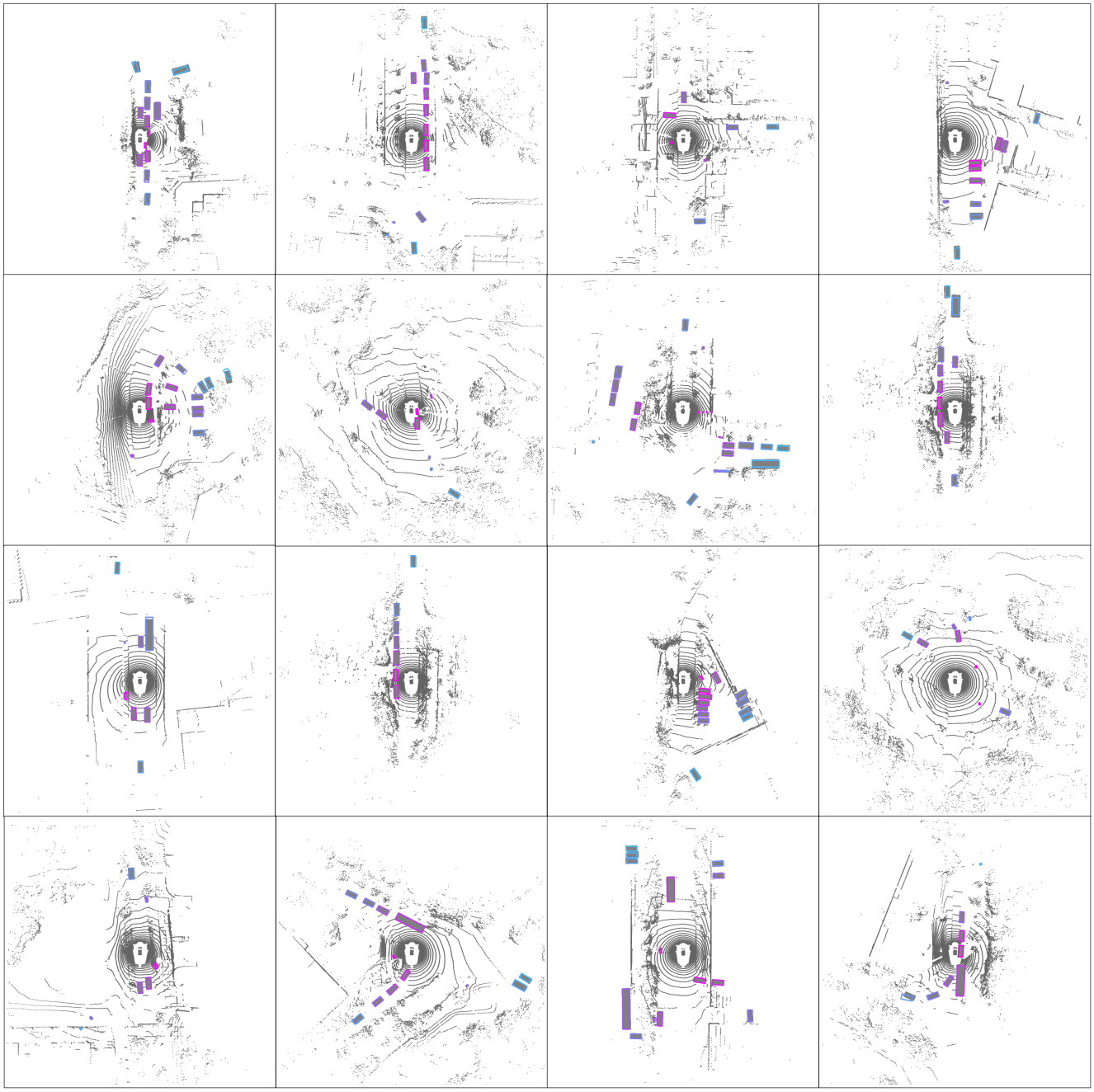

Figure A8. **Qualitative Results on Mamba-based Backbone.** We visualize bounding box generations from our method with Mamba-based backbone trained to generate boxes in near-to-far order. AutoReg3D generates boxes from first (magenta) to last (blue), ground-truth boxes are in gray. Best viewed in color.

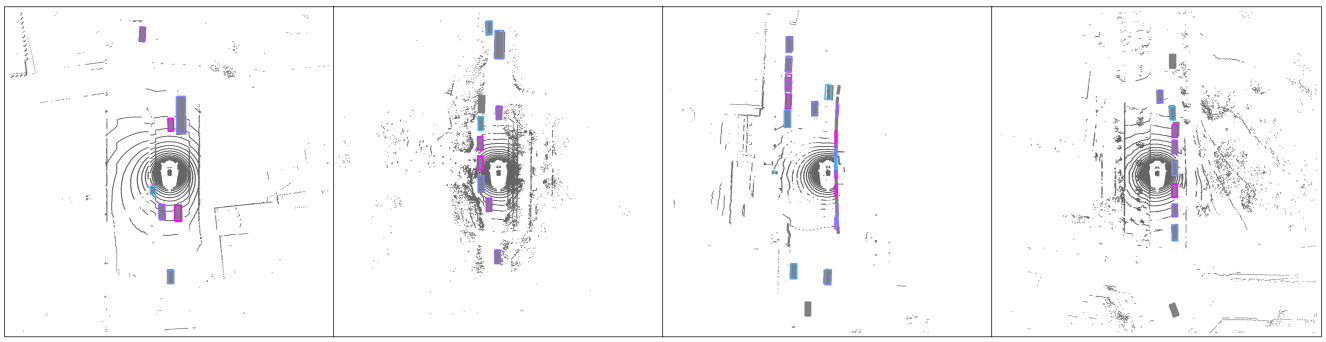

Figure A9. **Qualitative Results on Random-order Model.** We visualize bounding box generations from our method with voxel-based backbone trained to generate boxes in random order. AutoReg3D generates boxes from first (magenta) to last (blue), ground-truth boxes are in gray. Best viewed in color.

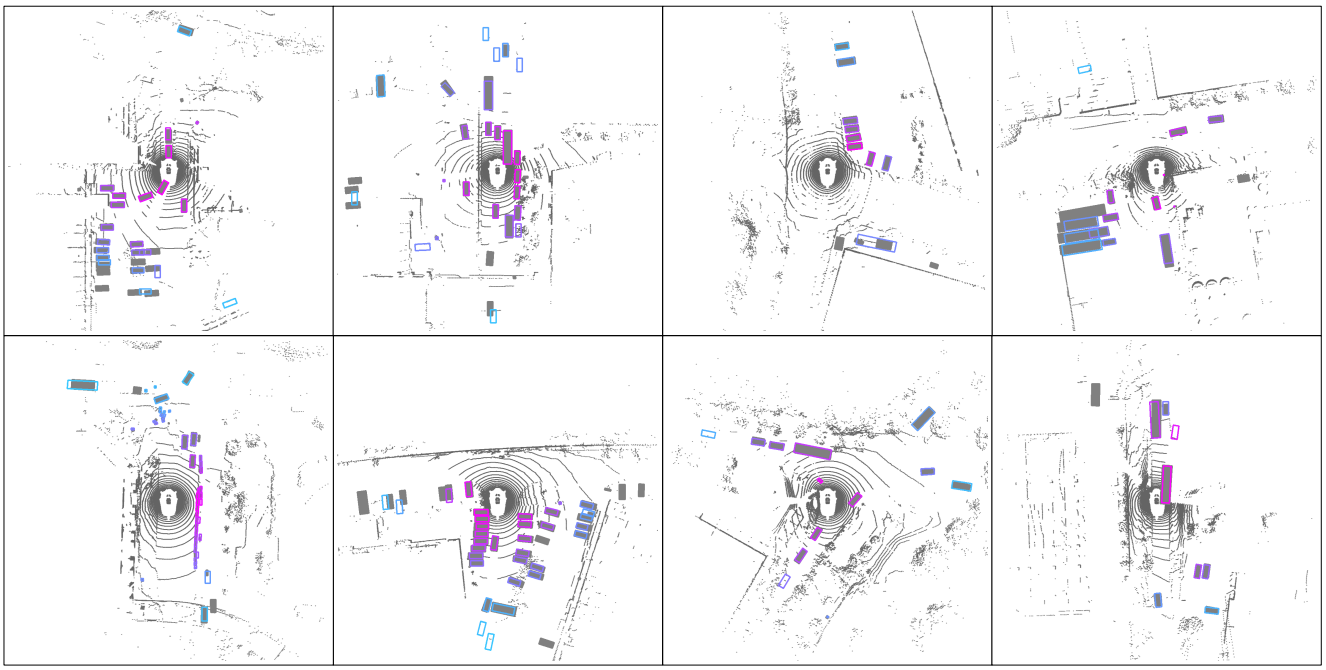

Figure A10. **Failure Case of AutoReg3D.** The dominant failure mode of our method include missing detections, extra detections, and misclassifications at distant locations or for sparse objects. AutoReg3D generates boxes from first (magenta) to last (blue), ground-truth boxes are in gray. Best viewed in color.