On the Feasibility and Opportunity of Autoregressive 3D Object Detection

Zanming Huang[1] Jinsu Yoo[1] Sooyoung Jeon[1] Zhenzhen Liu[2]
Mark Campbell[2] Kilian Q Weinberger[2] Bharath Hariharan[2] Wei-Lun Chao[1,3] Katie Z Luo[4]

[1]The Ohio State University [2]Cornell University [3]Boston University [4]Stanford University

Abstract

LiDAR-based 3D object detectors typically rely on proposal heads with hand-crafted components like anchor assignment and non-maximum suppression (NMS), complicating training and limiting extensibility. We present AutoReg3D, an autoregressive 3D detector that casts detection as sequence generation. Given point-cloud features, AutoReg3D emits objects in a range-causal (near-to-far) order and encodes each object as a short, discrete-token sequence consisting of its center, size, orientation, velocity, and class. This near-to-far ordering mirrors LiDAR geometry—near objects occlude far ones but not vice versa—enabling straightforward teacher forcing during training and autoregressive decoding at test time. AutoReg3D is compatible across diverse point-cloud or backbones and attains competitive nuScenes performance without anchors or NMS. Beyond parity, the sequential formulation unlocks language-model advances for 3D perception, including GRPO-style reinforcement learning for task-aligned objectives. These results position autoregressive decoding as a viable, flexible alternative for LiDAR-based detection and open a path to importing modern sequence-modeling tools into 3D perception.

1. Introduction

Object detection has long been one of the most influential tasks in computer vision, with applications spanning recognition [45–47], healthcare [34, 44], and analytics [5, 26, 55]. In 3D perception—critical for scene understanding and downstream tasks such as autonomous driving and robotic systems—detectors typically follow a "propose-then-classify" paradigm inspired by 2D methods [19]. Both two-stage variants [19, 47, 50–52] and single-stage counterparts [16, 45, 46] rely on classification and regression objectives to produce bounding boxes, achieving strong performance and remaining the dominant approach.

However, proposal-based detectors depend on a rigid stack of hand-crafted components—anchor assignment, proposal matching, geometric regression targets, confidence thresholds, and NMS [19, 23]. These arise because predictions are made independently across spatial locations, producing overlapping boxes that must be filtered and deduplicated. While effective, these components increase system complexity, complicate training, and often discard information during post-processing. They also hinder composability with downstream modules, *e.g.*, large language models (LLMs), limiting the scalability and extensibility of 3D detection beyond the original regimes.

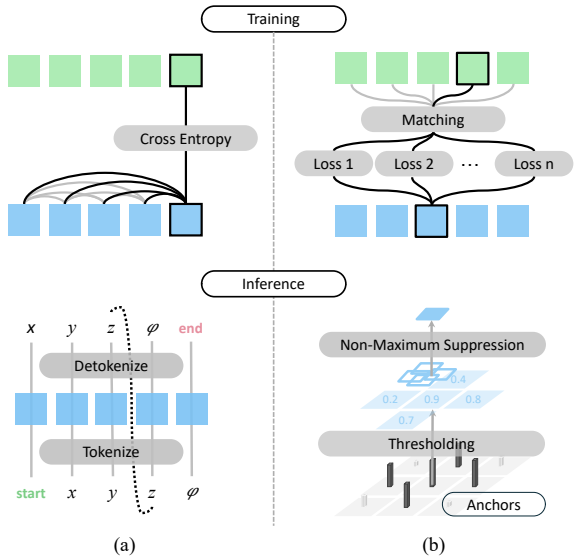


Figure 1. **Autoregressive Object Detection for 3D.** Our work proposes a 3D object detector that leverages a sequential generation representation (a). This eliminates many of the complications associated with the rigid detection pipeline, including anchor assignment, confidence thresholding, and NMS (b).

We address these limitations by reintroducing dependence among predictions. Inspired by the success of autoregressive (AR) sequence modeling in language [4, 53] and emerging vision-generation tasks [1, 33], we explore an alternative formulation for 3D detection: *casting bounding-box prediction as autoregressive sequence generation*. An AR model emits objects sequentially while conditioning

on previous outputs, naturally suppressing overlaps without NMS or proposal filtering. Although related ideas have shown promise in 2D detection [8, 9, 27], a scalable autoregressive approach for 3D point-cloud detection remains challenging due to the higher dimensionality, the difficulty of discretizing continuous geometry, and the spatial scale of LiDAR scenes.

We present AutoReg3D, the first autoregressive 3D point-cloud detector achieving performance competitive with proposal-based and query-based systems. Our key insight is that LiDAR naturally induces a near-to-far ordering: objects closer to the ego-vehicle are physically encountered—and thus observed—before distant ones. This ordering reflects the causal structure of occlusion in 3D and provides an intuitive *sequential dependence* along which objects can be generated autoregressively.

Given a point cloud encoded by any encoder (*e.g.*, voxel-based [63]), AutoReg3D generates a sequence of discrete tokens representing objects. Each object is encoded by tokens describing its class, location (x, y, z), size (l, w, h), velocity (v_x, v_y), and orientation ψ. Continuous geometry is discretized using ego-relative coordinates with parameter-specific vocabularies reflecting the ranges and semantics of each axis. Generation proceeds from a `[start]` token through near-to-far objects and terminates with `[end]`, producing a threshold-free set of boxes without confidence thresholds or NMS.

This formulation simplifies 3D detection while enabling techniques from sequence modeling, including reinforcement-learning (RL) fine-tuning [39, 40] and advanced decoding strategies [17, 21]. Our goal is to demonstrate viability: AR modeling can match mainstream detection accuracy while opening a path to modern sequence-modeling advances in 3D. Although sequential decoding introduces latency, we view speed as orthogonal to our core contribution and expect improvements as AR decoding and hardware continue to advance. Altogether, our contributions are:

- We introduce AutoReg3D, the first autoregressive 3D object detector that generates object sequences directly from point clouds, achieving performance comparable to leading proposal-based and query-based detectors.
- We demonstrate capabilities enabled by the autoregressive formulation, including elimination of NMS, compatibility with RL fine-tuning, and cascading refinement.

2. Related Works

3D Object Detection from Point Clouds. 3D object detection is a central task in autonomous driving and scene understanding, with methods developed for both camera inputs [7, 18, 45–47] and LiDAR point clouds [29, 36, 50, 63, 68]. LiDAR-based detectors typically process large point sets into deep features using point-based [10, 41, 42, 50, 64, 67], pillar-based [29, 51, 63, 68], or voxel-based [11, 13, 51, 52] backbones, with more recent transformer and Mamba-based architectures [37, 59, 70]. Despite these architectural advances, most detectors follow the propose-then-classify paradigm inherited from 2D detection [18, 19, 46]. Detection heads are typically implemented as two-stage [50, 51, 64] or one-stage [37, 59, 68] models that produce multiple independent predictions across spatial locations. These must be filtered using thresholds and suppression techniques such as NMS [19, 23, 32, 58, 65], introducing additional complexity and potential information loss.

Autoregressive Object Detection. Autoregressive sequence modeling has been widely studied in natural language processing and recently extended to vision tasks such as vision-language modeling [2, 25, 30, 33, 57]. Pix2Seq [8] first formulated 2D object detection as sequence prediction by generating bounding boxes and labels as discrete tokens, with later works extending this paradigm to unified 2D vision models [9, 27]. For 3D LiDAR point clouds, autoregressive detection remains largely unexplored. Point2Seq [62] predicts bounding-box attributes sequentially but generates boxes in parallel across bird's-eye-view cells. To the best of our knowledge, we are the first to investigate fully autoregressive detection for LiDAR point clouds.

3. Method

Existing regression-based 3D detectors rely heavily on hand-crafted components such as customized losses for each box attribute and anchor assignment strategies [50–52, 63]. Inspired by language modeling, we cast 3D object detection as a sequence generation task, demonstrating that such complexity can be largely avoided. Our framework simplifies the pipeline by adopting a unified cross-entropy objective and removing post-processing steps. Despite its simplicity, AutoReg3D achieves performance on par with state-of-the-art regression-based detectors.

By modeling a conditional distribution rather than performing direct regression, our formulation also unlocks two unique opportunities in improving detection performance: reinforcement fine-tuning and cascading refinement.

3.1. 3D Object Detection as Sequence Prediction

3D Object as Tokens. In LiDAR-based 3D object detection, objects are described by their class label and 3D bounding boxes parameterized by their center (x, y, z), dimensions (l, w, h), yaw angle (ψ), and velocity (v_x, v_y) in an ego-relative coordinate system. To represent bounding boxes using tokens, we are inspired by the quantization strategy of [8] and quantize the bounding box values uniformly into an integer between $t_k \in [1, n_k]$, where $k \in \{x, y, z, l, w, h, \psi, v_x, v_y\}$.

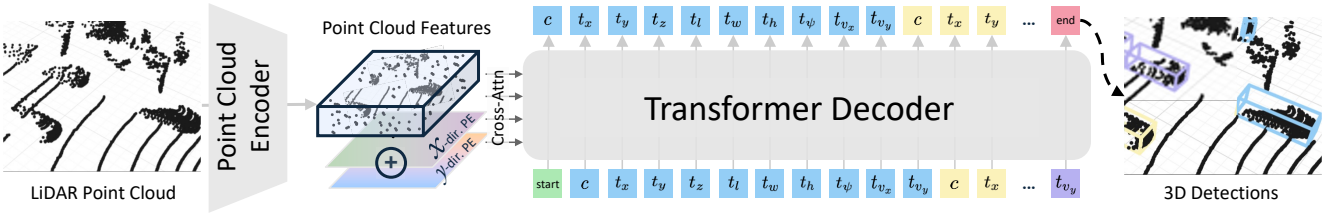


Figure 2. **Model Architecture.** We leverage an encoder-decoder architecture for encoding point cloud features, then generate tokenized bounding boxes with a causal Transformer decoder. We detokenize the generated sequence to obtain the final set of 3D object detections. This design is compatible with a variety of point cloud encoders, including pillar-, voxel-convolutions, transformer, and Mamba backbone.

Sequence Ordering. We order the token sequences of individual objects into a single sequence to represent the entire scene. Unlike 2D detection, objects in 3D scenes exhibit a natural depth ordering: nearer objects occlude farther ones. This intrinsic dependency in 3D object detection motivates the use of a deterministic ordering between objects in the scene. In particular, we arrange objects in *near-to-far* order according to their distance from the ego vehicle.

Autoregressive Model. We model the scene as a sequence of object tokens $\boldsymbol{o}$ generated autoregressively conditioned on the point cloud $\mathbf{X}$ and previously generated objects. To model the conditional probability $p(\boldsymbol{o}_i|\boldsymbol{o}_{1:i-1}, \mathbf{X})$, we adopt an encoder-decoder architecture. The encoder extracts the scene feature from the point cloud, while a Transformer decoder [56] autoregressively predicts tokens one at a time. We use a 6-layer Transformer decoder that cross-attends to point-cloud features from the backbone [29, 37, 63, 68]. This minimal design naturally captures dependencies among objects while remaining compatible with existing 3D encoders (Figure 2).

Training and Inference. We train the model to maximize the likelihood of the ground-truth token sequence $\boldsymbol{o}$ given the input point cloud $\mathbf{X}$. The optimization objective can be written as $\mathcal{L} = \sum_{i=1}^{N} \log P(\boldsymbol{o}_i|\boldsymbol{o}_{1:i-1}, \mathbf{X})$. At inference time, object tokens are sampled sequentially according to the learned conditional distribution $p(\boldsymbol{o}_i|\boldsymbol{o}_{1:i-1}, \mathbf{X})$. Our formulation allows the number of predicted objects to vary naturally and eliminates the need for anchors, confidence thresholds, or post-processing steps such as NMS.

3.2. Opportunities in Autoregressive 3D Detection

Reinforcement Learning Fine-tuning. While teacher-forcing maximizes token likelihood, it does not explicitly optimize the set-level detection objective. Our sequence prediction formulation enables RL fine-tuning of AutoReg3D with a sequence-level reward aligned with detection quality at inference time, thereby improving global consistency. We freeze the backbone and fine-tune AutoReg3D using GRPO [49] with a sequence-level reward based on the class-averaged F1 score between predicted and ground-truth boxes.

Method	Encoder	Det. Head	Prec.	Rec.	F1
PointPillars [29]	Pillar Conv.	Anchor-based	58.3	50.0	53.1
CenterPoint [68]		Center-based	67.9	53.3	59.5
Ours		AR Transformer	69.6	52.4	59.2
SECOND [63]	Voxel Conv.	Anchor-based	63.5	55.6	59.1
CenterPoint [68]		Center-based	72.8	60.3	65.8
Ours		AR Transformer	74.9	59.4	65.8
DSVT [59]	Transformer	Non-AR Transformer	79.1	66.3	71.6
Ours		AR Transformer	77.0	64.1	69.5
LION [37]	Mamba	Non-AR Transformer	78.6	68.3	72.5
Ours		AR Transformer	77.5	65.2	70.4

Table 1. **NuScenes Validation Detection Performance.** We compare AutoReg3D with baseline methods, grouped by encoder type. Our method achieves performance comparable to regression-based detectors across all backbones.

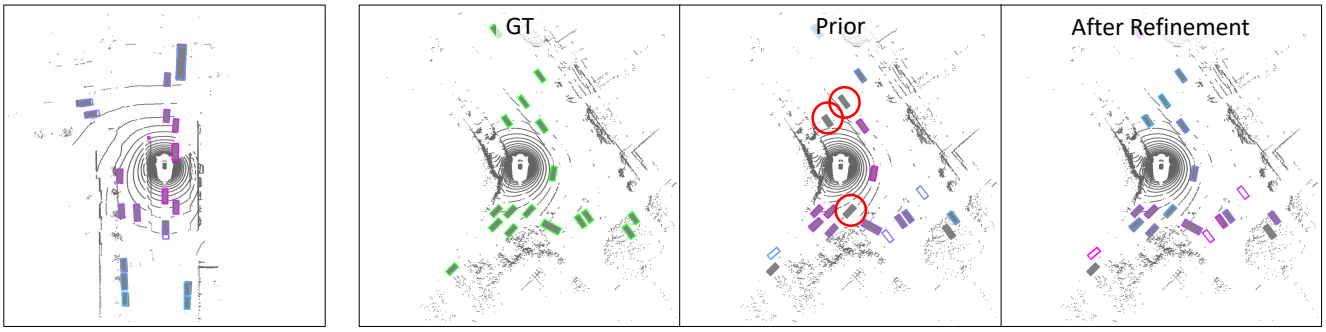
Model	Precision	Recall	F1
Teacher Forcing	**74.9**	59.4	65.8
+ GRPO	74.5	**60.9**	**66.7**

Table 2. **Performance with RL Fine-tuning.** Fine-tuning with GRPO directly on IoU further boosts performance.

Cascading Refinement. An advantage of our autoregressive formulation is its ability to incorporate external inputs as hints during inference. To illustrate, we cascade the output of a prior model, which generates a strong initial prediction, into a completion model that refines it. Predictions are aggregated via a simple strategy, yielding consistent improvements over using either model alone.

4. Experiments

We evaluate our method on nuScenes [6]. Standard 3D detection benchmarks report mean Average Precision (mAP). However, mAP requires detectors to output confidence scores, which originates from formulating detection as a classification problem, where the model predicts how likely each anchor corresponds to an object versus background. In contrast, our method casts detection as a sequence generation problem and does not produce such objectness scores. We therefore follow [28] and report Precision, Recall, and F1.



(a) (b)

Figure 3. **Qualitative Results.** (a) Autoregressive box generation. (b) Cascading refinement recovers missed objects (circled). Boxes generated from first (magenta) to last (blue), ground-truth boxes are in gray. Best viewed in color.

Model	Precision	Recall	F1
Prior only	**74.9**	59.4	65.8
Completion only	68.9	49.9	56.3
Prior → Completion	74.7	**60.2**	**66.2**

Table 3. **Cascading Refinement Performance.** Refinement via prompting the generation of a random-order model (Completion) improves performance over the distance-ordering model (Prior).

Method	Visibility Level			
	0–40%	40–60%	60–80%	80–100%
CenterPoint	28.9	30.5	43.2	67.0
Ours	30.0	31.1	42.7	67.6
Δ%	+4.1%	+1.8%	-1.3%	+0.8%

Table 4. **Performance with Occlusion.** We compare F1 scores between AutoReg3D and the baseline under different visibility levels. Autoregressive modeling improves detection, particularly under low visibility (heavy occlusion).

4.1. Main Results

Supervised Training Performance. In Table 1, we compare our method with representative baseline methods spanning four representative point cloud encoder families: pillar-based, voxel-based, transformer-based, and Mamba-based architectures. For each encoder type, we initialize our model with the corresponding pre-trained backbone used from prior works. For fair comparison, we select the confidence thresholds for all baseline detectors that maximizes F1 score on the *training* set. In Table 1, our method achieves performance on par with existing regression-based detectors across all encoder types. We observe higher precision for pillar and voxel models, which we attribute to the autoregressive conditioning between predicted boxes.

Reinforcement Learning Performance. Our autoregressive formulation also enables further performance gains through reinforcement learning. During RL fine-tuning, we freeze the encoder and optimize only the autoregressive detection head using GRPO. As shown in Table 2, this additional training stage improves the voxel-based model's F1 score from 65.8 to 66.7. The improvement is primarily driven by increased recall, reflecting the impact of our task-specific reward, which encourages the model to generate more complete prediction sequences and successfully detect objects that were previously missed.

4.2. Method Analysis

Results on Cascading Refinement. The near-to-far model provides strong initial detections but cannot recover missed boxes at arbitrary locations. A random-order completion model can generate boxes anywhere in the scene. Combining the two through conditional sampling improves performance over either model alone (Table 3).

Performance with Occlusion. We study the performance of AutoReg3D under occlusion by evaluating objects across four different visibility levels. Table 4 shows that AutoReg3D improves over the baseline *in cases of high occlusion* (visibility $\leq 60\%$), with the largest improvement at the most occluded case (0–40% visible). This supports the intuition that autoregressive modeling leverages the natural inter-object dependencies of 3D scenes, improving detection on partially observed objects.

5. Conclusion

This work provides the first viability study for autoregressive 3D object detectors, demonstrating competitive performance with proposal-based and query-based methods on standard benchmarks. By leveraging the natural ordering in LiDAR data and discretizing geometry with parameter-specific vocabularies, we show that the rigid detection pipeline can be replaced with a single autoregressive decoder. Our approach establishes that 3D detection can be formulated as sequence generation, connecting it to the broader ecosystem of autoregressive modeling advances. We hope this work encourages further exploration of sequence modeling for 3D perception tasks.

Acknowledgment

This work was supported by the National Artificial Intelligence Research Resource (NAIRR) Pilot and the Anvil supercomputer (award NSF-OAC 2005632). We thank the Ohio Supercomputer Center for providing computational resources. This work was funded in part by the National Science Foundation (IIS-2107077, IIS-2107161) and the New York Presbyterian Hospital.

References

[1] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. In *NeurIPS*, 2022. 1

[2] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *ICCV*, 2015. 2

[3] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond. *arXiv preprint arXiv:2308.12966*, 2023. 1

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *NeurIPS*, 2020. 1

[5] Matija Burić, Miran Pobar, and Marina Ivašić-Kos. Object detection in sports videos. In *MIPRO*, 2018. 1

[6] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020. 3

[7] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 2, 1

[8] Ting Chen, Saurabh Saxena, Lala Li, David J Fleet, and Geoffrey Hinton. Pix2seq: A language modeling framework for object detection. *arXiv preprint arXiv:2109.10852*, 2021. 2

[9] Ting Chen, Saurabh Saxena, Lala Li, Tsung-Yi Lin, David J. Fleet, and Geoffrey Hinton. A unified sequence interface for vision tasks. *arXiv preprint arXiv:2206.07669*, 2022. 2

[10] Xuesong Chen, Shaoshuai Shi, Benjin Zhu, Ka Chun Cheung, Hang Xu, and Hongsheng Li. Mppnet: Multi-frame feature intertwining with proxy points for 3d temporal object detection. In *ECCV*, 2022. 2

[11] Yukang Chen, Jianhui Liu, Xiangyu Zhang, Xiaojuan Qi, and Jiaya Jia. Voxelnext: Fully sparse voxelnet for 3d object detection and tracking. In *CVPR*, 2023. 2

[12] Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, et al. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv preprint arXiv:2412.05271*, 2024. 1

[13] Jiajun Deng, Shaoshuai Shi, Peiwei Li, Wengang Zhou, Yanyong Zhang, and Houqiang Li. Voxel r-cnn: Towards high performance voxel-based 3d object detection. In *AAAI*, 2021. 2

[14] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407, 2024. 1

[15] Ryan Ehrlich, Bradley Brown, Jordan Juravsky, Ronald Clark, Christopher Ré, and Azalia Mirhoseini. Codemonkeys: Scaling test-time compute for software engineering. *arXiv preprint arXiv:2501.14723*, 2025. 1

[16] Chengjian Feng, Yujie Zhong, Yu Gao, Matthew R Scott, and Weilin Huang. Tood: Task-aligned one-stage object detection. In *ICCV*, 2021. 1

[17] Markus Freitag and Yaser Al-Onaizan. Beam search strategies for neural machine translation. *arXiv preprint arXiv:1702.01806*, 2017. 2

[18] Ross Girshick. Fast r-cnn. In *ICCV*, 2015. 2

[19] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 1, 2

[20] Alex Graves. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*, 2012. 2

[21] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019. 2

[22] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *ICLR*, 2020. 2

[23] Jan Hosang, Rodrigo Benenson, and Bernt Schiele. Learning non-maximum suppression. In *CVPR*, 2017. 1, 2

[24] Yihan Hu, Zhuangzhuang Ding, Runzhou Ge, Wenxin Shao, Li Huang, Kun Li, and Qiang Liu. Afdetv2: Rethinking the necessity of the second stage for object detection from point clouds. In *AAAI*, 2022. 1

[25] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024. 2

[26] Deepak Kumar Jain et al. An evaluation of deep learning based object detection strategies for threat object detection in baggage security imagery. *PRL*, 2019. 1

[27] Qing Jiang, Junan Huo, Xingyu Chen, Yuda Xiong, Zhaoyang Zeng, Yihao Chen, Tianhe Ren, Junzhi Yu, and Lei Zhang. Detect anything via next point prediction. *arXiv preprint arXiv:2510.12798*, 2025. 2

[28] Qing Jiang, Junan Huo, Xingyu Chen, Yuda Xiong, Zhaoyang Zeng, Yihao Chen, Tianhe Ren, Junzhi Yu, and Lei Zhang. Detect anything via next point prediction. In *CVPR*, 2026. 3

[29] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019. 2, 3, 1

[30] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *ICML*, 2023. 2, 1

[31] Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Yu Qiao, and Jifeng Dai. Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. *arXiv preprint arXiv:2203.17270*, 2022. 1

[32] Feng Liang, Shaofei Yang, Tinrui Mai, and Yichen Yang. The design of objects bounding boxes non-maximum suppression and visualization module based on fpga. In *DSP*, 2018. 2

[33] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, 2023. 1, 2

[34] Yang Liu, Zhuo Ma, Ximeng Liu, Siqi Ma, and Kui Ren. Privacy-preserving object detection for medical images with faster r-cnn. *TIFS*, 2019. 1

[35] Yingfei Liu, Tiancai Wang, Xiangyu Zhang, and Jian Sun. Petr: Position embedding transformation for multi-view 3d object detection. In *ECCV*, 2022. 1

[36] Zhijian Liu, Haotian Tang, Alexander Amini, Xinyu Yang, Huizi Mao, Daniela Rus, and Song Han. Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation. *arXiv preprint arXiv:2205.13542*, 2022. 2

[37] Zhe Liu, Jinghua Hou, Xinyu Wang, Xiaoqing Ye, Jingdong Wang, Hengshuang Zhao, and Xiang Bai. Lion: Linear group rnn for 3d object detection in point clouds. In *NeurIPS*, 2024. 2, 3

[38] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. s1: Simple test-time scaling. In *EMNLP*, 2025. 1

[39] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *NeurIPS*, 2022. 2

[40] André Susano Pinto, Alexander Kolesnikov, Yuge Shi, Lucas Beyer, and Xiaohua Zhai. Tuning computer vision models with task rewards. In *ICML*, 2023. 2

[41] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 2

[42] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017. 2

[43] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*, 2020. 1

[44] Mohammed Gamal Ragab, Said Jadid Abdulkadir, Amgad Muneer, Alawi Alqushaibi, Ebrahim Hamid Sumiea, Rizwan Qureshi, Safwan Mahmood Al-Selwi, and Hitham Alhussian. A comprehensive systematic review of yolo for medical object detection (2018 to 2023). *IEEE Access*, 2024. 1

[45] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *CVPR*, 2017. 1, 2

[46] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016. 1, 2

[47] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015. 1, 2

[48] Danila Rukhovich, Anna Vorontsova, and Anton Konushin. Fcaf3d: Fully convolutional anchor-free 3d object detection. In *ECCV*, 2022. 1

[49] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. 3, 4

[50] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *CVPR*, 2019. 1, 2

[51] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *CVPR*, 2020. 2, 1

[52] Shaoshuai Shi, Li Jiang, Jiajun Deng, Zhe Wang, Chaoxu Guo, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn++: Point-voxel feature set abstraction with local vector representation for 3d object detection. *IJCV*, 2023. 1, 2

[53] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023. 1

[54] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. In *NeurIPS*, 2017. 1

[55] Adam Van Etten. You only look twice: Rapid multi-scale object detection in satellite imagery. *arXiv preprint arXiv:1805.09512*, 2018. 1

[56] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 3

[57] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *CVPR*, 2015. 2

[58] Paul Viola and Michael J Jones. Robust real-time face detection. *IJCV*, 2004. 2

[59] Haiyang Wang, Chen Shi, Shaoshuai Shi, Meng Lei, Sen Wang, Di He, Bernt Schiele, and Liwei Wang. Dsvt: Dynamic sparse voxel transformer with rotated sets. In *CVPR*, 2023. 2, 3

[60] Tai Wang, Xinge Zhu, Jiangmiao Pang, and Dahua Lin. Fcos3d: Fully convolutional one-stage monocular 3d object detection. In *ICCV*, 2021. 1

[61] Yue Wang, Vitor Campagnolo Guizilini, Tianyuan Zhang, Yilun Wang, Hang Zhao, and Justin Solomon. Detr3d: 3d object detection from multi-view images via 3d-to-2d queries. In *CoRL*, 2022. 1

[62] Yujing Xue, Jiageng Mao, Minzhe Niu, Hang Xu, Michael Bi Mi, Wei Zhang, Xiaogang Wang, and Xinchao Wang. Point2seq: Detecting 3d objects as sequences. In *CVPR*, 2022. 2

[63] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 2018. 2, 3, 1

[64] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *CVPR*, 2018. 2

[65] Bo Yang, Jianan Wang, Ronald Clark, Qingyong Hu, Sen Wang, Andrew Markham, and Niki Trigoni. Learning object bounding boxes for 3d instance segmentation on point clouds. In *NeurIPS*, 2019. 2

[66] Chenyu Yang, Yuntao Chen, Haofei Tian, Chenxin Tao, Xizhou Zhu, Zhaoxiang Zhang, Gao Huang, Hongyang Li, Y. Qiao, Lewei Lu, Jie Zhou, and Jifeng Dai. Bevformer v2: Adapting modern image backbones to bird's-eye-view recognition via perspective supervision. In *CVPR*, 2023. 1

[67] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Std: Sparse-to-dense 3d object detector for point cloud. In *ICCV*, 2019. 2

[68] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *CVPR*, 2021. 2, 3, 1

[69] Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? In *NeurIPS*, 2025. 4

[70] Guowen Zhang, Lue Fan, Chenhang He, Zhen Lei, Zhaoxiang Zhang, and Lei Zhang. Voxel mamba: Group-free state space models for point cloud based 3d object detection. In *NeurIPS*, 2024. 2

[71] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *CVPR*, 2018. 1

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 analysis.
- 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 [3, 14]. Additionally, while we leverage a semantically expanded vocabulary, learned codebooks [54] 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 [12, 15, 38] that trade inference compute for performance may apply to detection. Furthermore, autoregressive representations could facilitate integration with language [14, 43, 53] and vision-language models [3, 30, 33]. 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 [29, 51, 63, 71] 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 [68] pioneered this by detecting object centers as keypoints in bird's-eye view before regressing to 3D boxes, while FCOS3D [60] extended this to monocular detection. AFDetV2 [24] and FCAF3D [48] 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 [7] leverage transformers with learnable object queries. DETR3D [61] introduced sparse queries that predict 3D boxes through cross-attention with multi-view features, while PETR [35] added position-embedded queries and BEVFormer [31, 66] 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 [33]. We conducted preliminary experiments with a decoder-only formulation using a Q-Former [30] 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.

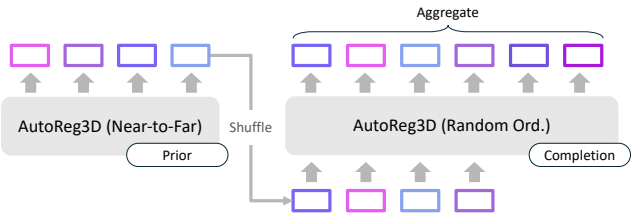


Figure A1. **Cascading Refinement Methodology.**

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 3 of the main text. Additional qualitative results for Cascading Refinement are provided in Figure A5 and Figure A6.

C. Additional Analysis

C.1. Effect of Object Ordering

We study how different object ordering strategies affect detection performance in our autoregressive framework. As shown in Table A1, we compare three orderings: random order, descending order by the number of LiDAR points within each ground-truth box, and a distance-based near-to-far ordering. We observe that the near-to-far ordering significantly outperforms both random and point number-based ordering. By predicting objects from near to far, the model effectively exploits the interdependent nature of the task, where close-by objects inform the prediction of farther objects. Ordering by point count performs better than random ordering, as it partially correlates with distance (closer objects often contain more points), but it remains an imperfect proxy because small yet close objects (e.g., pedestrians) may have fewer points. Random ordering performs the worst, as it fails to exploit any structural dependencies between objects.

C.2. Effect of Token Ordering

We also ablate the ordering of tokens within each object sequence. As shown in Table A2, we compare three strategies: placing the class token at the beginning, in the middle (after box location), or at the end of the sequence. Both the class-first and class-middle orderings outperform the class-last variant, with class-first yielding the best results. This indicates that having the model determine the object class earlier in the sequence provides useful context for predicting the remaining attributes, thereby improving overall detection performance.

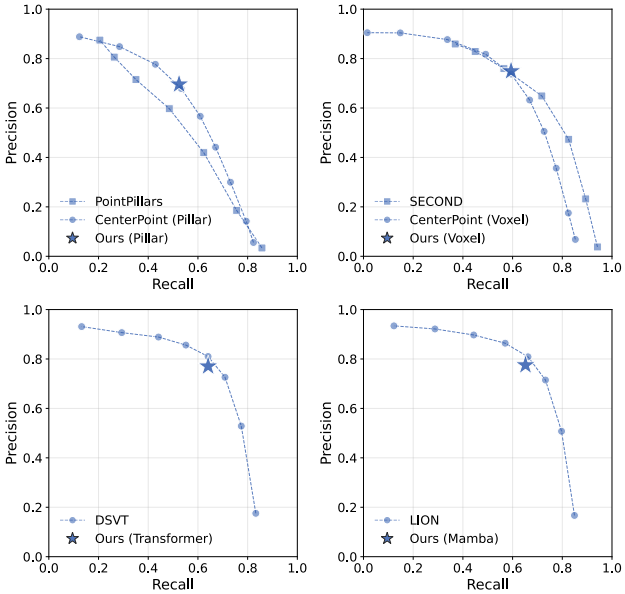


Figure A2. **Precision-Recall Plot.** We plot the PR curves for the baseline methods, and the precision-recall point using our autoregressive decoder with a star. *Top left*: Pillar-based backbone. *Top right*: Voxel-based backbone. *Bottom left:* Transformer-based backbone. *Bottom right:* Mamba-based backbone. We observe that the precision-recall point of AutoReg3D consistently hits or lies outside the PR curves of models with the same backbone.

C.3. Effect of Decoding Method

We evaluate how different decoding strategies influence detection performance, as shown in Table A3. Specifically, we compare nucleus sampling [22] (top-p=0.95, top-k=50), greedy decoding, and beam search [20] (4 beams). Beam search achieves the best performance by trading off inference time for accuracy, as it explores multiple candidate sequences and approximates a more globally optimal decoding trajectory, whereas greedy decoding selects only the most likely next token. Nucleus sampling performs the worst, as it favors diversity over accuracy, which is essential for precise object detection.

C.4. Ordering Comparison by Distance

We analyze the performance differences between the random-order model and the near-to-far model. As shown in Table A4, the near-to-far model outperforms the random-

Order	Precision	Recall	F1
Random	68.9	49.9	56.3
Point Number	72.8	55.2	61.8
Near-to-far	**74.9**	**59.4**	**65.8**

Table A1. **Ablation on Object Ordering.**

Order	Precision	Recall	F1
Cls. Last	74.4	58.1	64.9
Cls. Middle	74.8	58.4	65.2
Cls. First	**74.9**	**59.4**	**65.8**

Table A2. **Ablation on Token Ordering.**

Method	Precision	Recall	F1
Nucleus	67.1	57.8	61.9
Greedy	74.9	59.4	65.8
Beam Search	**75.0**	**59.9**	**66.1**

Table A3. **Ablation on Decoding Method.**

	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 A4. **Ordering Comparison by Distance to Ego**. Near-to-far order outperforms random order across all metrics and distance bins.

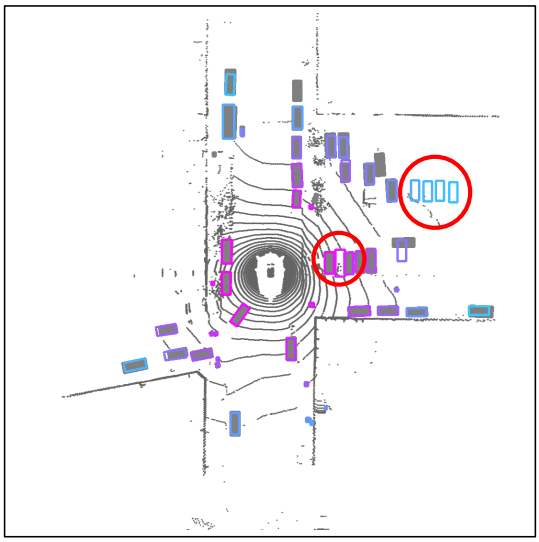


Figure A3. **Failure Case Example.**Predicted boxes are colored by generation order from first (magenta) to last (blue). Best viewed in color.

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.5. Failure Mode Analysis

Similar to existing detectors, AutoReg3D's failure modes include missed detections, extra detections, and misclassifications, especially for distant or sparse objects. Interestingly, the extra boxes predicted by AutoReg3D generally respect the overall scene structure (Figure 3 (c)): they rarely overlap with existing predictions and tend to follow road geometry. In contrast to methods that model objects independently, the autoregressive formulation leverages object inter-relations by conditioning on previous predictions, allowing AutoReg3D to produce plausible "guesses" grounded in scene geometry.

C.6. 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 A5, 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.

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 A4.

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

Table A5. **Expanded *vs*. Shared Vocabulary.**

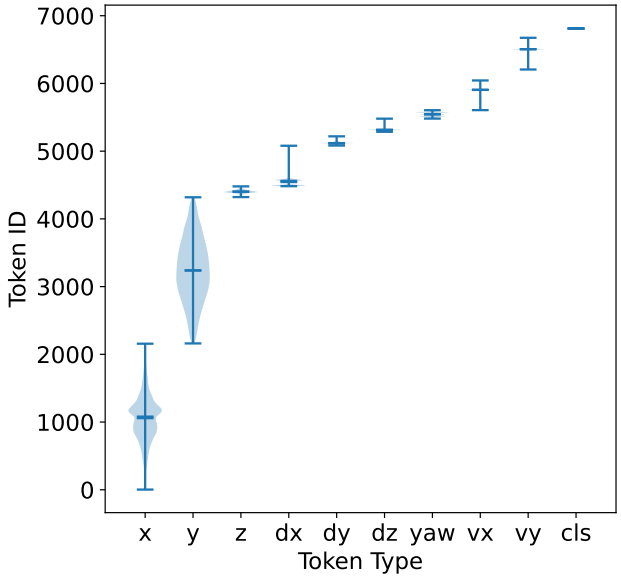


Figure A4. **Token Distribution After Quantization.** Violin plot of the token distribution, plotting token ID by type. Semantic types occupy disjoint ID ranges, with types that cover larger value ranges correspondingly taking up larger token ranges.

To justify our choice of token resolution and bin count, we ablate different settings in Table A7. 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 errors, 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.7. RL Decoding Methods

We compare the effect of different decoding strategies on our method after RL fine-tuning in Table A6. Similar to teacher-forcing in Table A3, nucleus sampling (F1=65.8) performs worse than greedy (F1=66.7) and beam search

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 A6. **Analysis on RL Decoding Method.** Different generation decoding strategies have limited effect on performance after RL fine-tuning.

(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 [69]. This sharpening enables the RL model to outperform teacher-forcing (F1=61.9) under nucleus sampling.

D. Additional Details on AutoReg3D

D.1. Reinforcement Learning Fine-tuning

We adopt RL strategy based on GRPO [49]. Given a scene point cloud $\mathbf{X}$, we sample a group of G detection sequences $\{\boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_G\}$ from the current detection policy π_θ.

We define a task-aligned reward. Given a set of ground-truth boxes $B_c = \{b_1, b_2, \ldots, b_3\}$ and a set predicted boxes $\hat{B}_c = \{\hat{b}_1, \hat{b}_2, \ldots, \hat{b}_3\}$ that belong to class $c \in \mathcal{C}$, we calculate each ground-truth box's maximum IoU r_i^* with the predicted boxes in the same class. We calculate the class-averaged reward r inspired by F1:

$$r_c^{\text{Rec.}} = \frac{1}{|B_c|} \sum_{i=1}^{|B_c|} r_i^*, \quad r_c^{\text{Prec.}} = \frac{1}{|\hat{B}_c|} \sum_{i=1}^{|B_c|} r_i^*,$$
$$r = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \frac{2\, r_c^{\text{Prec.}}\, r_c^{\text{Rec.}}}{r_c^{\text{Prec.}} + r_c^{\text{Rec.}}}. \tag{A1}$$

The GRPO objective is formulated as follows:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \frac{1}{G} \sum_{i=1}^{G} \frac{1}{|\boldsymbol{o}_i|} \sum_{t=1}^{|\boldsymbol{o}_i|} \Big[\min\Big(\rho_{i,t} \hat{A}_{i,t},\, \text{clip}(\rho_{i,t}, 1-\epsilon, 1+\epsilon) \hat{A}_{i,t}\Big) - \beta\, \mathbb{D}_{\text{KL}}\big[\pi_\theta \,\|\, \pi_{\text{ref}}\big] \Big] \tag{A2}$$

Where $\hat{A}_{i,t}$ is the estimated advantage, $\rho_{i,t}$ is the importance sampling ratio, and π_{ref} is the frozen detection policy after supervised training.

D.2. 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.

D.3. Pseudocode of AutoReg3D

We show the pseudocode of AutoReg3D training and inference. Our autoregressive formulation largely simplifies the modeling process compared to proposal-based and query-based detectors.

Algorithm 1 Training (Teacher Forcing)

```
# Enc: point cloud encoder
# Dec: autoregressive Transformer decoder
# cross_entropy(): unified CE loss over all tokens

for X, boxes in dataloader: # point cloud and GT boxes
    F = Enc(X)
    boxes = sort(boxes) # enforce ordering

    seq = tokenizer.encode(boxes)
    seq_in  = seq[:-1]
    seq_tgt = seq[1:]

    logits = Dec(F, seq_in)
    loss = cross_entropy(logits, seq_tgt)
    loss.backward()
```

Algorithm 2 Inference (Autoregressive Decoding)

```
# Enc: point cloud encoder
# Dec: autoregressive Transformer decoder

def detect(X):
    F = Enc(X)
    seq = ["<START>"]
    while True:
        logits = Dec(F, seq)
        t = argmax(logits[-1])  # greedy decoding
        seq.append(t)
        if t == "<END>" or len(seq) > T_max:
            break
    boxes = tokenizer.decode(seq) # tokens to 3D boxes
    return boxes
```

E. Additional Quantitative Results

We show additional nuScenes quantitative results in Table A8, listing the threshold used for each baseline and TP errors of each method.

F. Additional Qualitative Results

We include additional qualitative results:

- Figure A5 & Figure A6: Qualitative results on Cascading Refinement.
- Figure A7: Qualitative results on pillar-based backbone.
- Figure A8: Qualitative results on voxel-based backbone.
- Figure A9: Qualitative results on transformer-based backbone.
- Figure A10: Qualitative results on Mamba-based backbone.
- Figure A11: Qualitative results on random-order model.
- Figure A12: 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 A7. **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 A8. **Detailed Evaluation Result.**

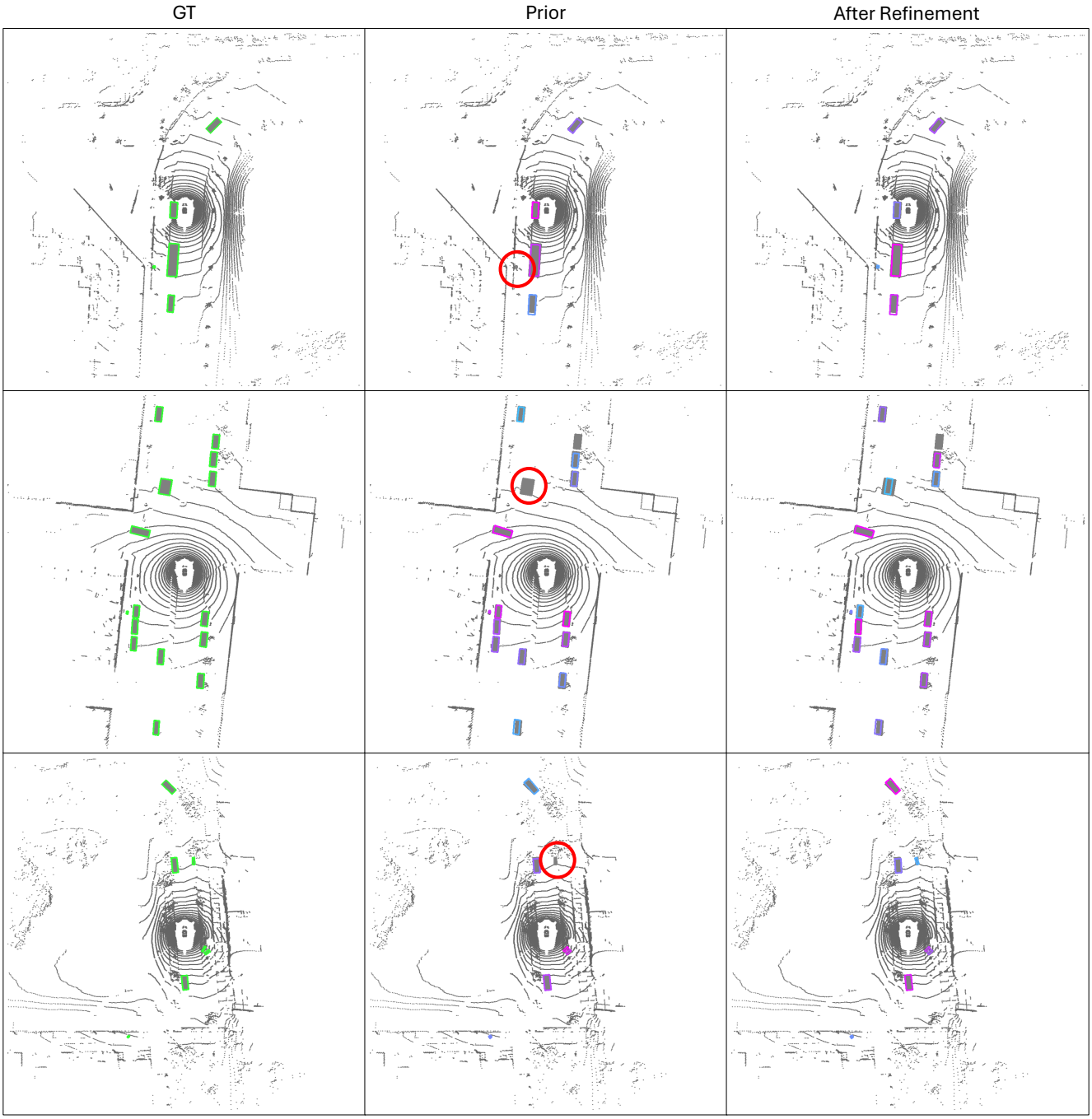


Figure A5. **Qualitative Results on Cascading Refinement.** We visualize the ground-truth boxes (left), predictions from the near-to-far prior model (middle), and predictions after Cascading Refinement (right). Cascading Refinement recovers objects that were missed by the prior model (circled in red). Predicted boxes are colored by generation order from first (magenta) to last (blue). Best viewed in color.

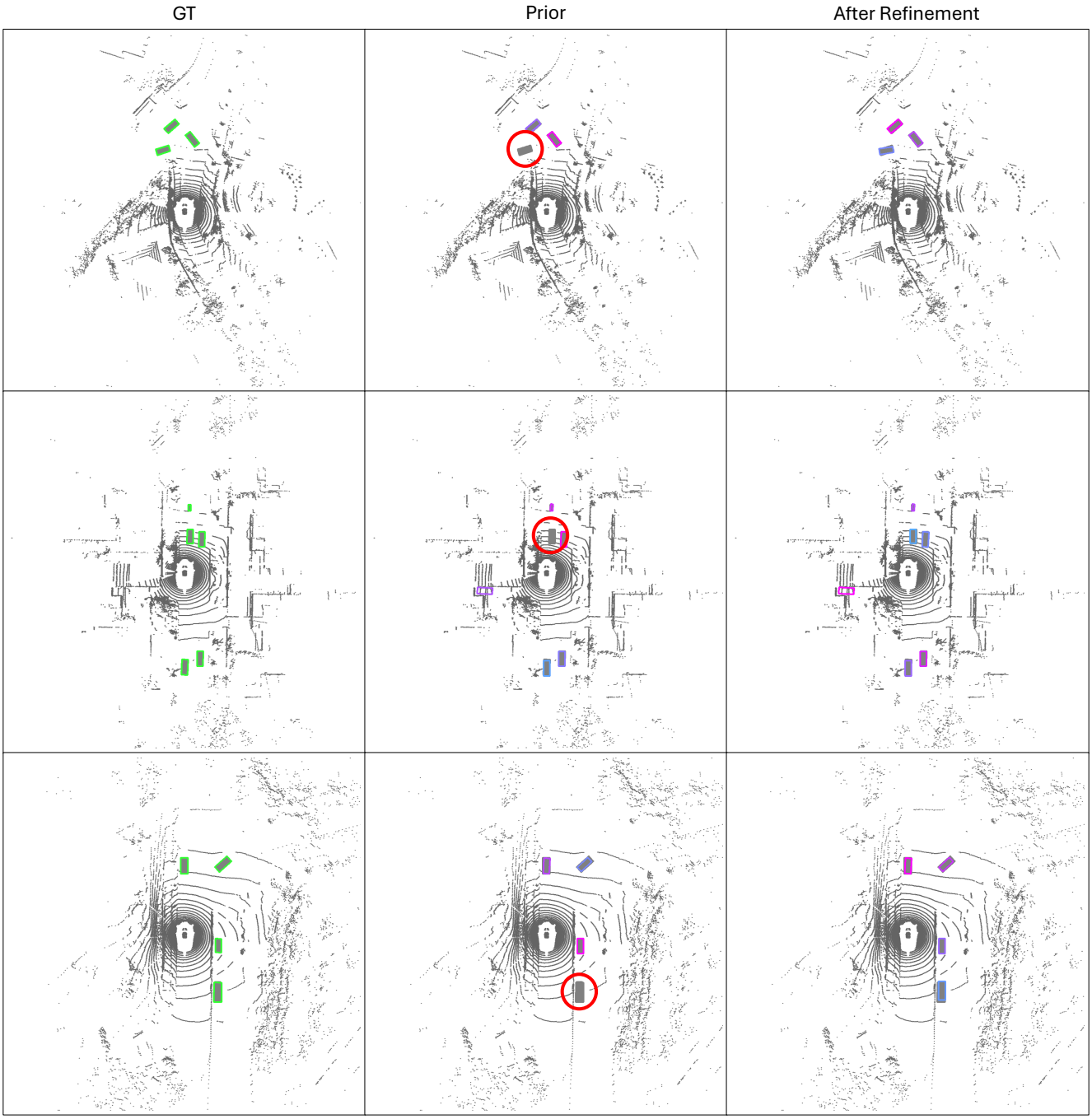


Figure A6. **Qualitative Results on Cascading Refinement.** We visualize the ground-truth boxes (left), predictions from the near-to-far prior model (middle), and predictions after Cascading Refinement (right). Cascading Refinement recovers objects that were missed by the prior model (circled in red). Predicted boxes are colored by generation order from first (magenta) to last (blue). Best viewed in color.

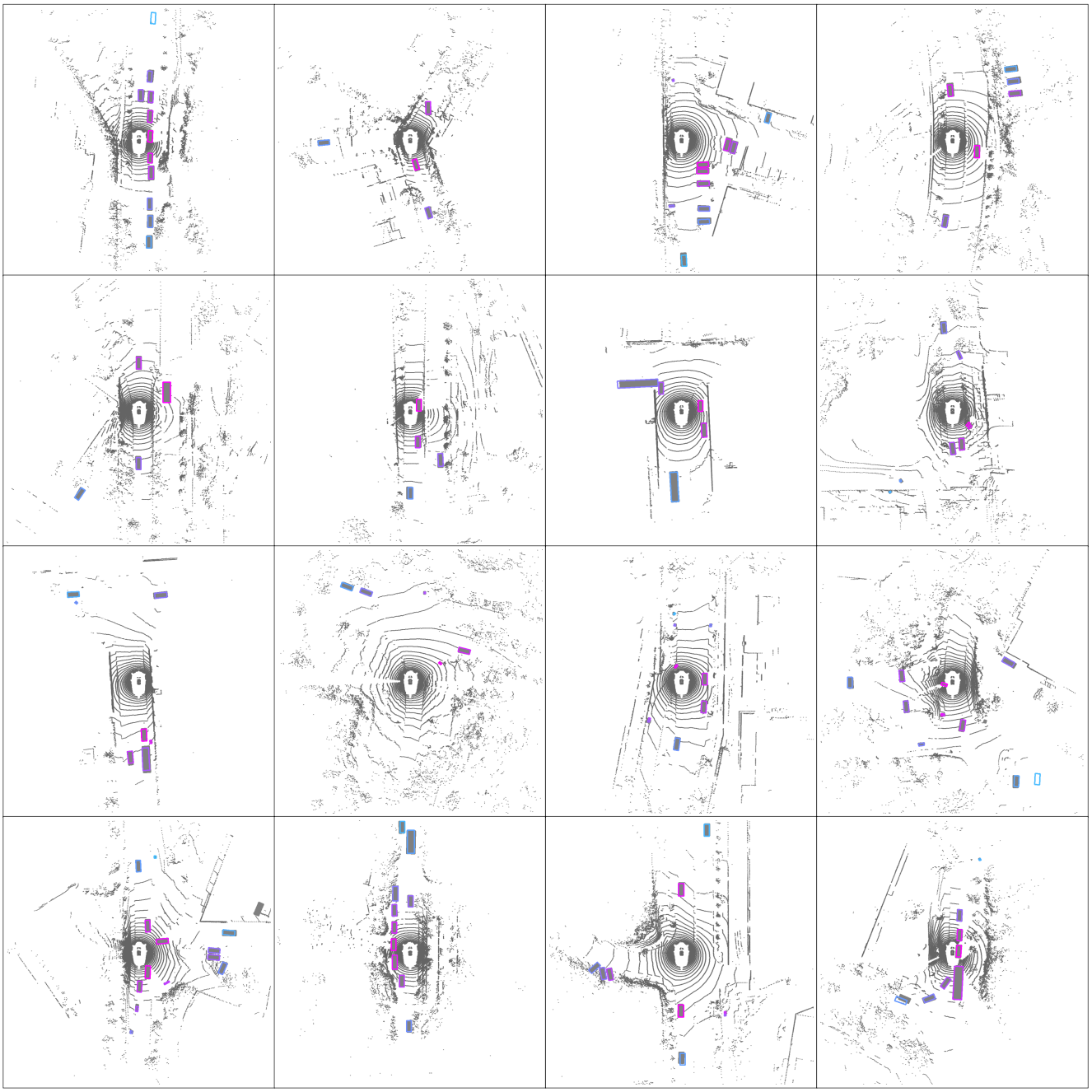


Figure A7. **Qualitative Results on Pillar-based Backbone.** We visualize bounding box generations from our method with pillar-based backbone trained to generate boxes in near-to-far order. AutoReg3D generates boxes from first (magenta) to last (blue), ground-truth boxes are in gray. Best viewed in color.

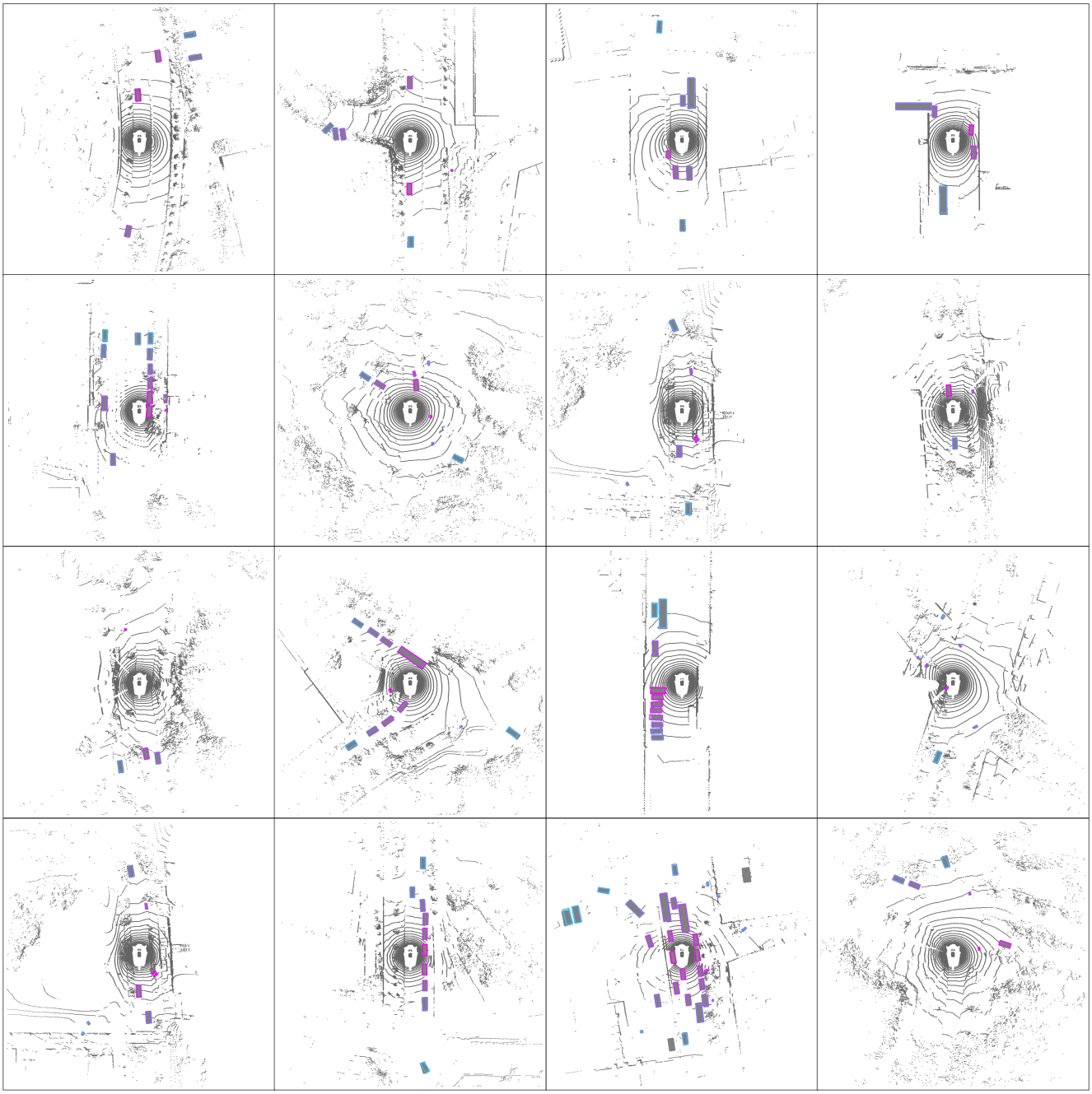


Figure A8. **Qualitative Results on Voxel-based Backbone.** We visualize bounding box generations from our method with voxel-based backbone trained to generate boxes in near-to-far order. AutoReg3D generates boxes from first (magenta) to last (blue), ground-truth boxes are in gray. Best viewed in color.

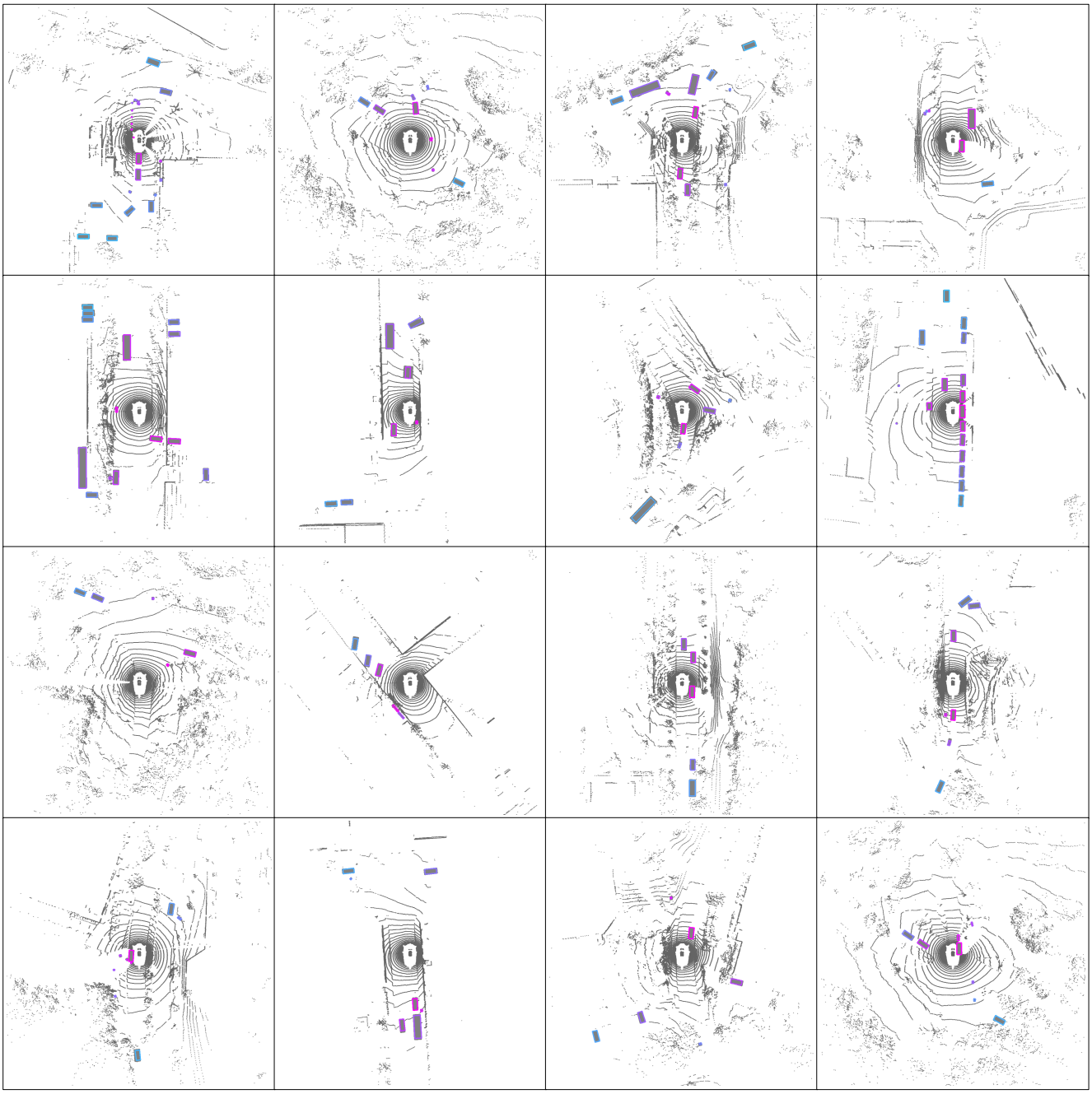


Figure A9. **Qualitative Results on Transformer-based Backbone.** We visualize bounding box generations from our method with transformer-based backbone trained to generate boxes in near-to-far order. AutoReg3D generates boxes from first (magenta) to last (blue), ground-truth boxes are in gray. Best viewed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

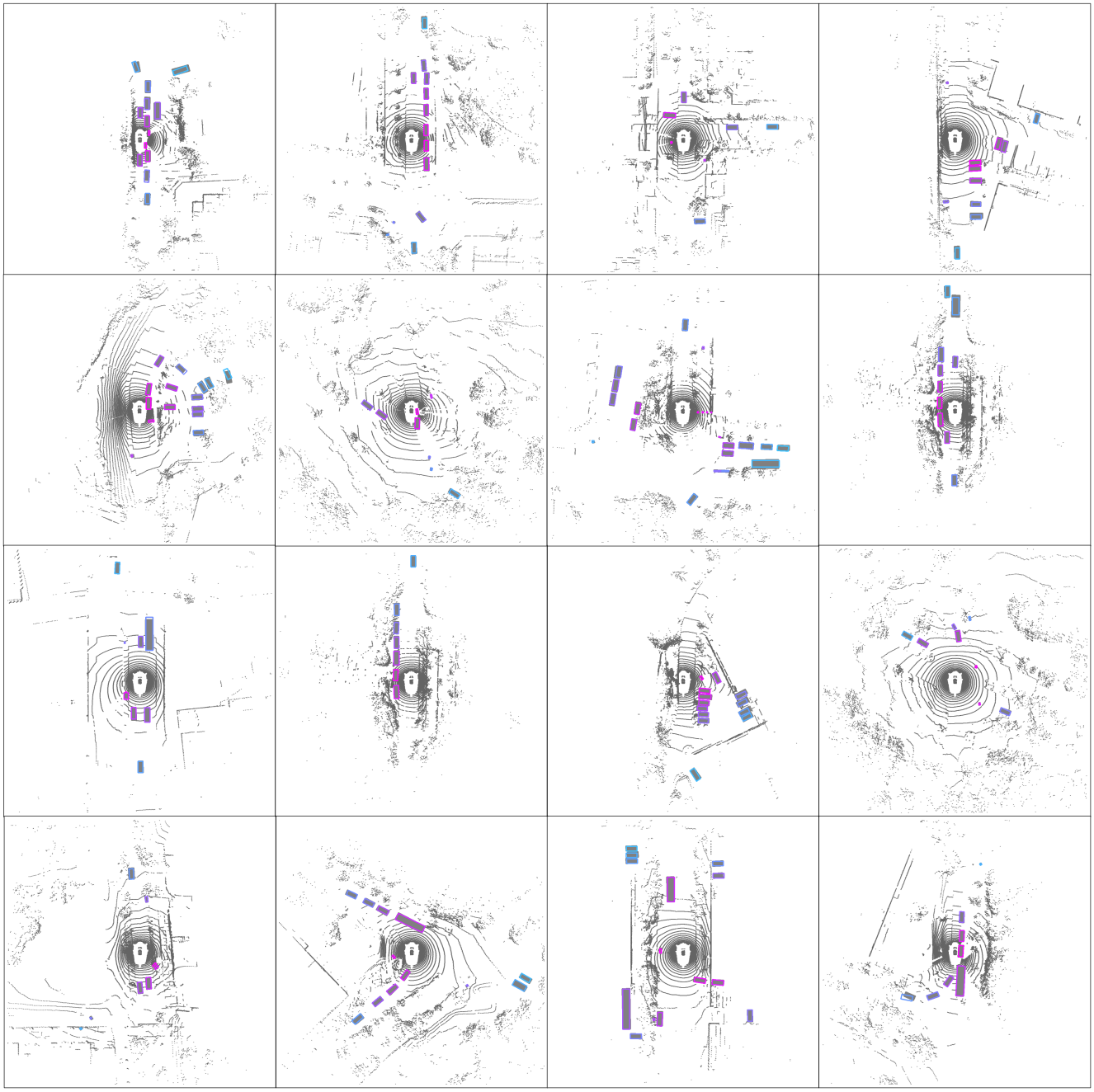


Figure A10. **Qualitative Results on Mamba-based Backbone.** We visualize bounding box generations from our method with Mamba-based backbone trained to generate boxes in near-to-far order. AutoReg3D generates boxes from first (magenta) to last (blue), ground-truth boxes are in gray. Best viewed in color.

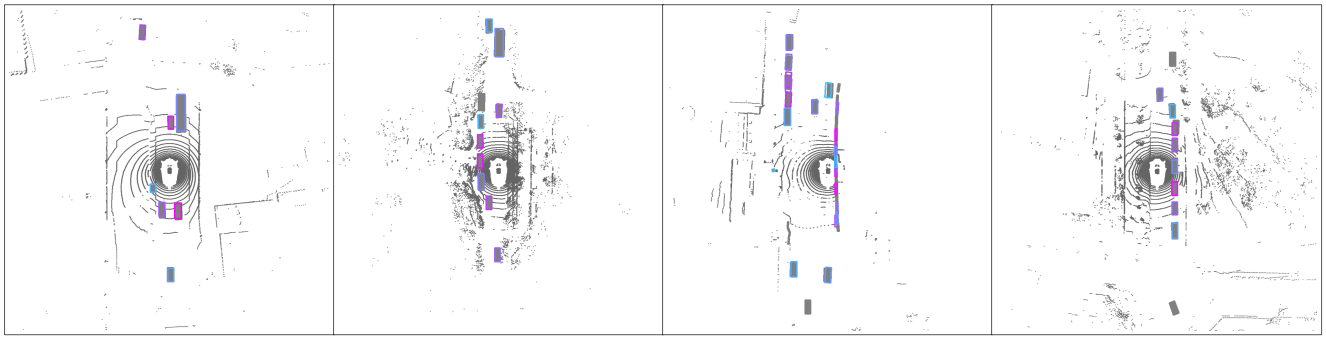


Figure A11. **Qualitative Results on Random-order Model.** We visualize bounding box generations from our method with voxel-based backbone trained to generate boxes in random order. AutoReg3D generates boxes from first (magenta) to last (blue), ground-truth boxes are in gray. Best viewed in color.

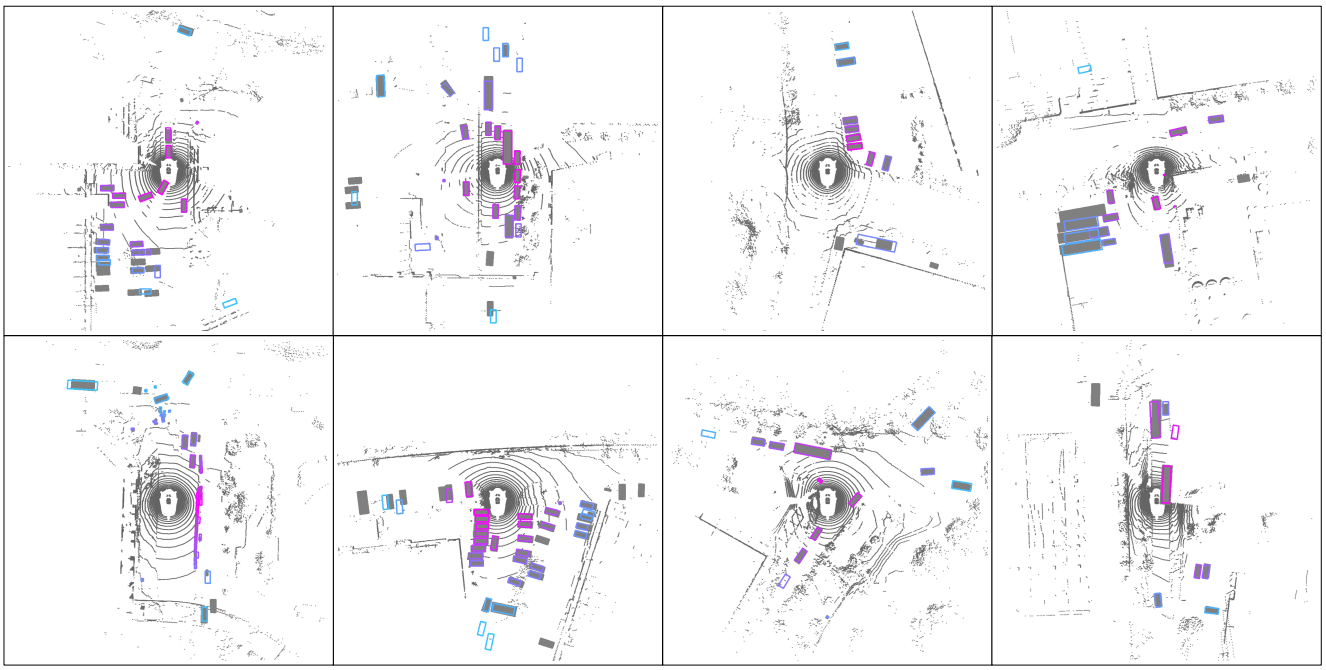


Figure A12. **Failure Case of AutoReg3D.** The dominant failure mode of our method include missing detections, extra detections, and misclassifications at distant locations or for sparse objects. AutoReg3D generates boxes from first (magenta) to last (blue), ground-truth boxes are in gray. Best viewed in color.