

LabelFormer: Auto-labelling Objects from LiDAR Point Clouds via Trajectory-level Refinement

Anonymous Author(s)

Affiliation

Address

email

Abstract: In these supplementary materials, we provide implementation details for our method and experiments, additional experimental results, and qualitative examples of our approach compared to the baselines. Please refer to the supplementary video for a visual presentation.

1 Implementation and Experiment Details

1.1 Model Details

Point Encoder: The point encoder is consisted of a voxelizer followed by a CNN-based backbone and a Feature Pyramid Network (FPN) [1].

Specifically, the voxelizer employs voxel resolution of 10cm in all of X , Y and Z directions, with a region of interest of $[-12, 12]$ meters along X , $[-4, 4]$ meters along Y , and $[-0.2, 3.0]$ meters along Z , to construct a $N_x \times N_y \times N_z$ voxel grid with $N_x = 240$, $N_y = 80$ and $N_z = 320$. For all points in each 3D voxel grid, we first represent each point as $(\Delta x, \Delta y, \Delta z, \Delta t)$ where $(\Delta x, \Delta y, \Delta z)$ is the positional offset with respect to the voxel centroid and $\Delta t = t - t_{ref}$ is the difference between the per-point time and the LiDAR sweep end time of the middle frame in the object trajectory. We feed this four-vector representation of each point into a two-layer MLP with 16 output channels each, and LayerNorm [2] and ReLU applied right after the first layer. Then, for each voxel, we pool all point features inside by summing them and applying a LayerNorm after to derive voxel grid features $\mathbb{R}^{N_x \times N_y \times N_z \times 16}$, which can be viewed as $N_x \times N_y$ “feature pillars” along the Z axis. We additionally encode an z -axis positional embedding via a learnable variable block $\in \mathbb{R}^{N_z \times 16}$. We concatenate the non-empty voxels in each feature pillar with the positional embedding block to obtain an augmented feature $\in \mathbb{R}^{N'_z \times 32}$ ($N'_z \leq N_z$ is the number of non-empty voxels in the pillar), and pass through a two-layer MLP with 16 and 32 output channels each (with LayerNorm and ReLU in between), apply LayerNorm after the second layer, and sum all the features along each pillar to obtain a BEV feature map $\in \mathbb{R}^{N_x \times N_y \times 32}$.

The backbone takes the BEV feature map as input and first applied three stem layers with 120, 96, 96 output channels each. Each stem layer is consisted of a 3x3 convolutional layer, followed by GroupNorm (GN) and ReLU. The first stem layer has stride of 2 while the next two have a stride of 1. Then, the output then passes through three downsampling stages, with each stage containing 6, 6, 4 ResNet [3] blocks with 288, 384, 576 output channels respectively. Each ResNet block applies a sequence of 1x1 conv, GN, ReLU, 3x3 conv (with an optional stride parameter), GN, ReLU, 1x1 conv, GN, ReLU to obtain a residual and sum with the input. Each stage first downsamples the input with a 1x1 conv block with stride 2 followed by GN, and the applies the first ResNet block with stride 2 in the middle 3x3 conv. The remaining ResNet blocks in each stage all have stride of 1.

The FPN then takes the outputs from all three stages in the backbone, which are $4\times$, $8\times$ and $16\times$ downsampled from the original resolution, and fuses the two lowest resolution feature maps first by applying a 1x1 conv block + GN to the $16\times$ low-resolution map, upsampling it by $2\times$ with bilinear interpolation, and adding it to the $8\times$ downsampled feature map. We then perform a similar

operation to fuse the $4\times$ downsampled feature map with the new fused $8\times$ downsampled feature map, and apply a final 3×3 conv to output a feature map of $4\times$ downsampled original resolution with channel dimension 256.

Attention Block: We next provide more details on the feed-forward MLP in each attention block. The feed-forward MLP is consisted of a linear layer with input dimension 256 and output dimension 512, followed by ReLU, DropOut with 10%, a second linear layer with input dimension 512 and output dimension 256 and another DropOut with 10%. We add the output of the MLP to the input of the MLP and return the sum.

Training Loss Details: In this section, we provide detailed definitions for our loss functions. The regression loss is defined as:

$$L_{reg}(\{\hat{\mathbf{b}}_i\}, \{\mathbf{b}_i^*\}) = \frac{\lambda}{M} \sum_i \text{smooth}\ell_1(\hat{x}_i, x_i^*) + \text{smooth}\ell_1(\hat{y}_i, y_i^*) + \text{smooth}\ell_1(\hat{l}, l^*) + \text{smooth}\ell_1(\hat{w}, w^*) \\ + \frac{1}{M} \sum_i \text{smooth}\ell_1(\sin(\hat{\theta}_i), \sin(\theta_i^*)) + \text{smooth}\ell_1(\cos(\hat{\theta}_i), \cos(\theta_i^*)) , \quad (1)$$

with the hyperparameter $\lambda = 0.1$ in practice, and the IoU loss is given by:

$$L_{IoU}(\{\hat{\mathbf{b}}_i\}, \{\mathbf{b}_i^*\}) = \frac{1}{M} \sum_i \text{IoU}(\text{BBox}(\hat{x}_i, \hat{y}_i, \hat{l}, \hat{w}), \text{BBox}(x_i^*, y_i^*, l^*, w^*)) , \quad (2)$$

to compare the axis-aligned refined and ground-truth bounding box in each frame.

1.2 Detector and Tracker

To obtain the first-stage coarse initialization, we follow the standard “detect-then-track” approach where a detection model is trained to output per-frame detections and we leverage a tracker to obtain consistent tracklets over time. Next, we give more details about the detector and tracker we use.

Detector: To boost detection performance, we adapt the single-frame public implementation of both PointPillars [4] and VoxelNeXt [5] to a multi-frame version that additionally takes 4 past history frames and 4 future frames as input. The validation mean AP of the single-frame vs. multi-frame PointPillars models are 68.78% and 71.02% on the Highway dataset respectively, and 55.98% and 60.58% on AV2 respectively. The validation mean AP of single-frame vs. multi-frame VoxelNeXt are 81.87%/84.25% on Highway and 60.06%/66.25% on AV2.

Tracker: Following [6, 7], we use a simple online tracker, which is largely inspired by [8], and we provide the implementation details of our tracker, in particular how association is performed across frames.

For each new frame at time step t with detections $\mathbf{B}_t = \{\mathbf{b}_t^l\}$ where each $\mathbf{b}_t^l = (x_t^l, y_t^l, l_t^l, w_t^l, \theta_t^l) \in \mathbb{R}^5$ is the individual 2D BEV bounding box, we first filter with Non-Maximum Suppression with IoU threshold 0.1, and then filter out bounding boxes with low confidence scores. We then compute a cost matrix with existing tracklets $\mathbf{S}_t = \{\mathbf{s}_t^j\}$ as follows. For each tracklet j , we first predict its bbox position (x_t^j, y_t^j) at time t : if the tracklet has at least two past frames, we set $(x_t^j, y_t^j) = 2 * (x_{t-1}^j, y_{t-1}^j) - (x_{t-2}^j, y_{t-2}^j)$ via naive extrapolation (assuming constant velocity between two adjacent frames); otherwise we simply set $(x_t^j, y_t^j) = (x_{t-1}^j, y_{t-1}^j)$. Then, for each pair of the detected bbox \mathbf{b}_t^l and the predicted tracklet bbox \mathbf{b}_t^j , we compute the Euclidean distance between the bbox centroids as $\ell^{j,l} = \sqrt{(x_t^j - x_t^l)^2 + (y_t^j - y_t^l)^2}$. For each existing tracklet, we simply employ a greedy strategy to find the nearest detection $l^* = \arg \min_l \ell^{j,l}$, and if the closest distance ℓ^{j,l^*} is greater than a threshold of 5.0m, then the tracklet has no match. We use greedy matching instead of a more sophisticated matching strategy such as Hungarian matching because it is more robust to noisy and spurious detections.

Motion State	First-Stage Detector	Second-Stage Refinement	Mean IoU	RC @ 0.5	RC @ 0.6	RC @ 0.7	RC @ 0.8
Stationary	PointPillars	-	64.46	79.34	69.95	53.58	28.66
		Auto4D	67.51	82.56	74.66	61.26	36.91
		3DAL	68.00	81.35	75.04	63.26	41.93
		Ours (LabelFormer)	70.67	84.08	77.31	66.03	46.81
	VoxelNeXt	-	68.91	83.53	75.15	61.53	38.81
		Auto4D	70.87	86.11	79.06	66.28	42.86
		3DAL	70.63	84.55	77.74	66.50	45.13
		Ours (LabelFormer)	73.21	86.77	80.09	69.54	51.08
Dynamic	PointPillars	-	60.53	73.12	61.04	43.23	21.03
		Auto4D	63.26	76.05	65.56	50.27	27.41
		3DAL	60.80	72.70	60.03	45.36	22.12
		Ours (LabelFormer)	65.64	78.06	68.69	54.74	34.10
	VoxelNeXt	-	62.25	73.96	62.17	46.66	25.57
		Auto4D	64.73	77.01	66.77	52.02	30.72
		3DAL	63.33	75.85	64.60	48.55	26.18
		Ours (LabelFormer)	66.93	79.49	69.64	56.04	35.88

Table 1: [AV2] Performance break-down for ground-truth stationary vs. dynamic objects

76 If a tracklet is matched to a new detection, we add the detection to the tracklet and update the tracklet
77 score $c_t^j = \frac{w \cdot c_{t-1}^j + c_t^l}{w + 1.0}$, where c_{t-1}^j is the old tracklet score, $c_t^l = 1.0$ is the detection confidence score
78 we set for every new detection, and $w = \sum_{i=1}^{n_{t-1}^j} 0.9^i$ where n_{t-1}^j is the number of tracking steps in
79 the tracklet.

80 If a tracklet is not matched, we grow the tracklet by naively extrapolating the position and angle,
81 and set the new confidence score as $c_t^j = 0.9c_{t-1}^j$.

82 If a new detection is not matched to any tracklet, we start a new tracklet and initialize the confidence
83 score c_t^j with the detection's confidence.

84 We terminate all tracklets with a tracking confidence score less than 0.1, and apply NMS at the end
85 over all existing tracklets in the current frame with an IoU threshold of 0.1. We repeat this process
86 for the next frame at time $t + 1$ until the end of the sequence.

87 1.3 Association with GT Trajectories

88 For each initial object trajectory detected and tracked in the first stage, we use a simple heuristic
89 to associate it with a ground-truth object trajectory as follows: for each frame that the detected
90 trajectory is present, we identify the ground-truth bounding box that has the maximum IoU with the
91 detected bounding box in that frame. If such ground-truth box has IoU less than 10%, then we fail
92 to find a matching ground-truth box for this frame. As a result we obtain M' ground-truth object
93 IDs for a detected trajectory of length M , with $0 \leq M' \leq M$ as we might not be able to find a
94 ground-truth ID for every frame. If M' is 0, then we have failed to find an associated ground-truth
95 object: we consider the detected object as a false positive and discard it in trajectory refinement
96 training and evaluation. Otherwise we take the most common ground-truth actor id out of the M'
97 objects and assign it as the associated ground-truth object trajectory for training and evaluation.

98 2 Additional Experiments

99 **Static vs. Dynamic Objects** The AV2 validation set contains around 52% stationary objects (we
100 classify an actor as static if the max displacement in the ground-truth displacement in all X , Y and Z
101 direction is within 1.0m). Table 1 additionally shows the dynamic vs. stationary object break-down
102 on the AV2 dataset. Our method is able to achieve significantly higher refinement accuracy on both
103 static and dynamic objects with a single network.

104 **Ablation with PointPillars Init:** We additionally performed the same set of ablation studies as
105 Table 3 in the main paper on the Highway dataset with the PointPillars-based initializations. Table 2

	Bbox Enc.	Point Enc.	Perturb	Window	# Att	Mean IoU	RC@0.5	RC@0.6	RC@0.7	RC@0.8
M_1	✓		✓	All	3	64.20	69.83	59.62	46.11	26.58
M_2		✓	✓	All	3	69.56	82.36	75.84	64.72	44.50
M_3	✓	✓		All	3	68.91	80.84	73.05	61.65	42.89
M_4	✓	✓	✓	5	3	69.07	81.19	74.17	63.47	45.05
M_5	✓	✓	✓	10	3	69.33	81.83	74.80	64.12	45.42
M_6	✓	✓	✓	All	3	70.59	83.09	75.77	65.11	46.16
M_7	✓	✓	✓	All	6	70.93	83.50	76.51	66.34	47.85

Table 2: **[Highway] Ablation study** using the PointPillars initializations. Perturb refers to the bounding box perturbation augmentation, Window specifies the window size when applicable, and # Att is the number of self-attention blocks.

Positional Encoding	Mean IoU	RC @ 0.5	RC @ 0.6	RC @ 0.7	RC @ 0.8
Absolute [10]	71.71	83.63	76.97	67.11	48.67
AliBi [9]	72.38	83.68	77.45	68.33	50.06

Table 3: **[Highway] Ablation of positional encoding** using the VoxelNeXt initializations

shows the results, which give the same conclusions as the VoxelNeXt-based initializations in the main paper.

Positional Encoding We additionally ablate our choice of positional encoding with the VoxelNeXt initialization on the Highway dataset. Table 3 shows that the relative positional encoding AliBi [9] gives overall better performance than using the vanilla absolute positional encodings [10].

3 Qualitative Results

In this section we show qualitative results for trajectory refinement, comparing LabelFormer with the coarse initialization, 3DAL [7] and Auto4D [6].

We illustrate initial and refined auto-labels for the Highway dataset with VoxelNeXt initializations. Fig. 1 showcases trajectories of two objects on the top that have sparse observations (and hence worse initializations) at the beginning, and denser observations towards the end, and ours LabelFormer is able to give better refinement for the worse initializations because it is able to leverage more temporal context more effectively than previous works. Fig. 1, 2, 3, 4 additionally showcase that our method works better qualitatively on trajectories with both sparse and dense observations and with various speeds. For more visualizations on Argoverse, please refer to the supplementary video.

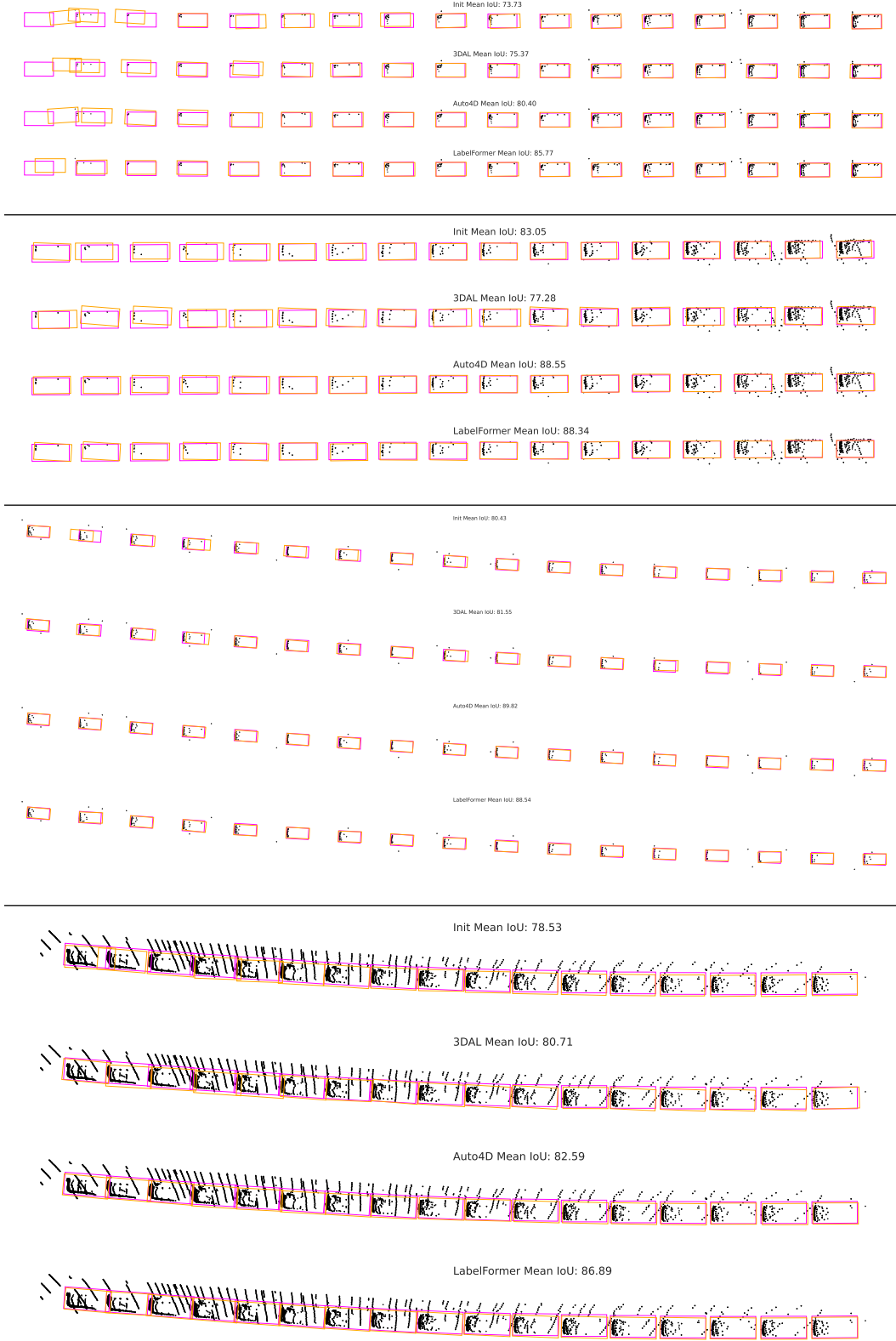


Figure 1: **[Highway] Qualitative results** showcasing different object trajectories (first-stage init, refined by 3DAL, Auto4D and Ours LabelFormer) in each object's trajectory coordinate frame. The ground-truth bounding box is in magenta, and the auto-label is in orange. To avoid cluttering, we visualize every other three bounding box in the first 50 frames of the trajectory.

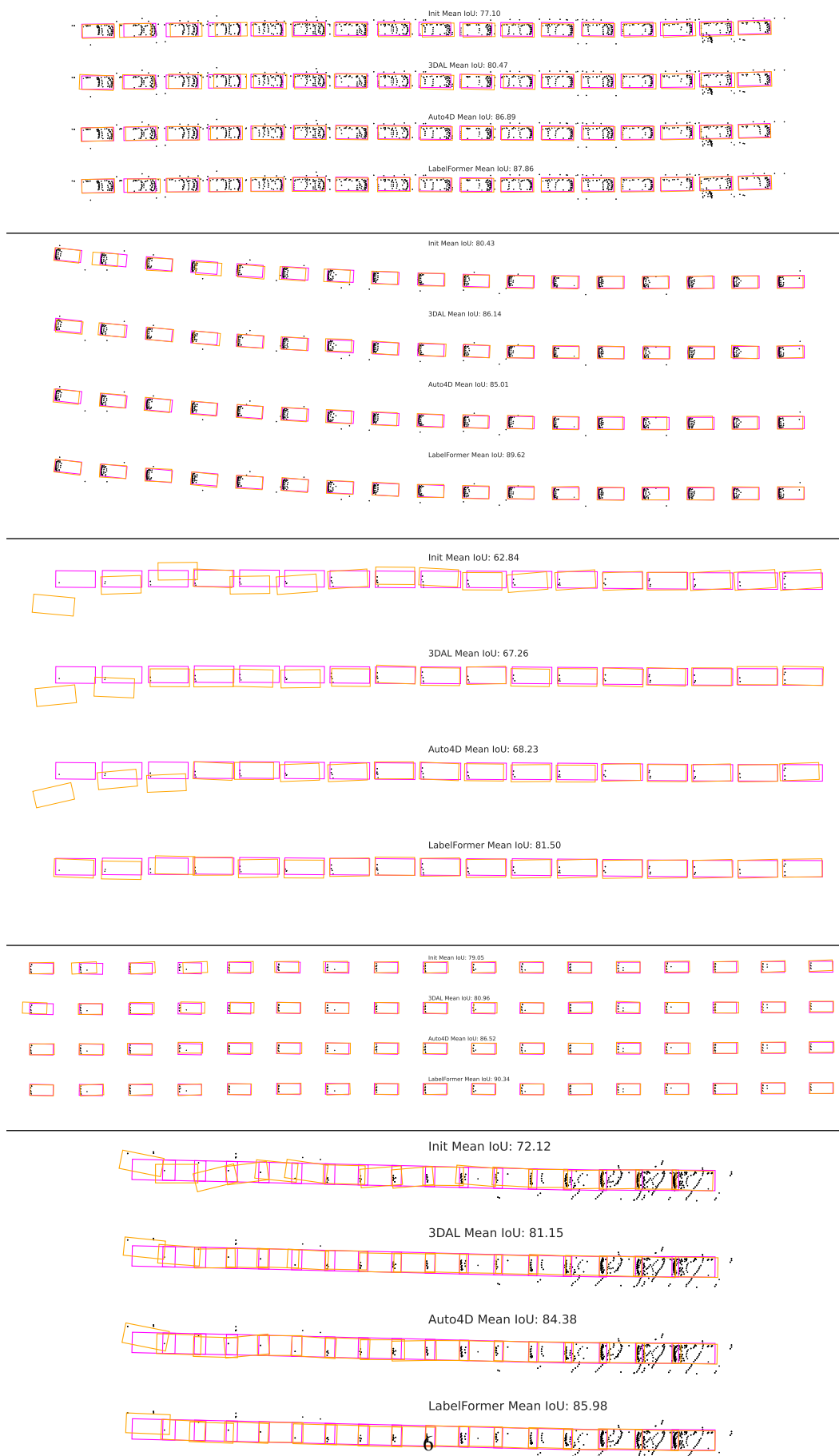


Figure 2: [Highway] More Qualitative results



Figure 3: [Highway] More Qualitative results

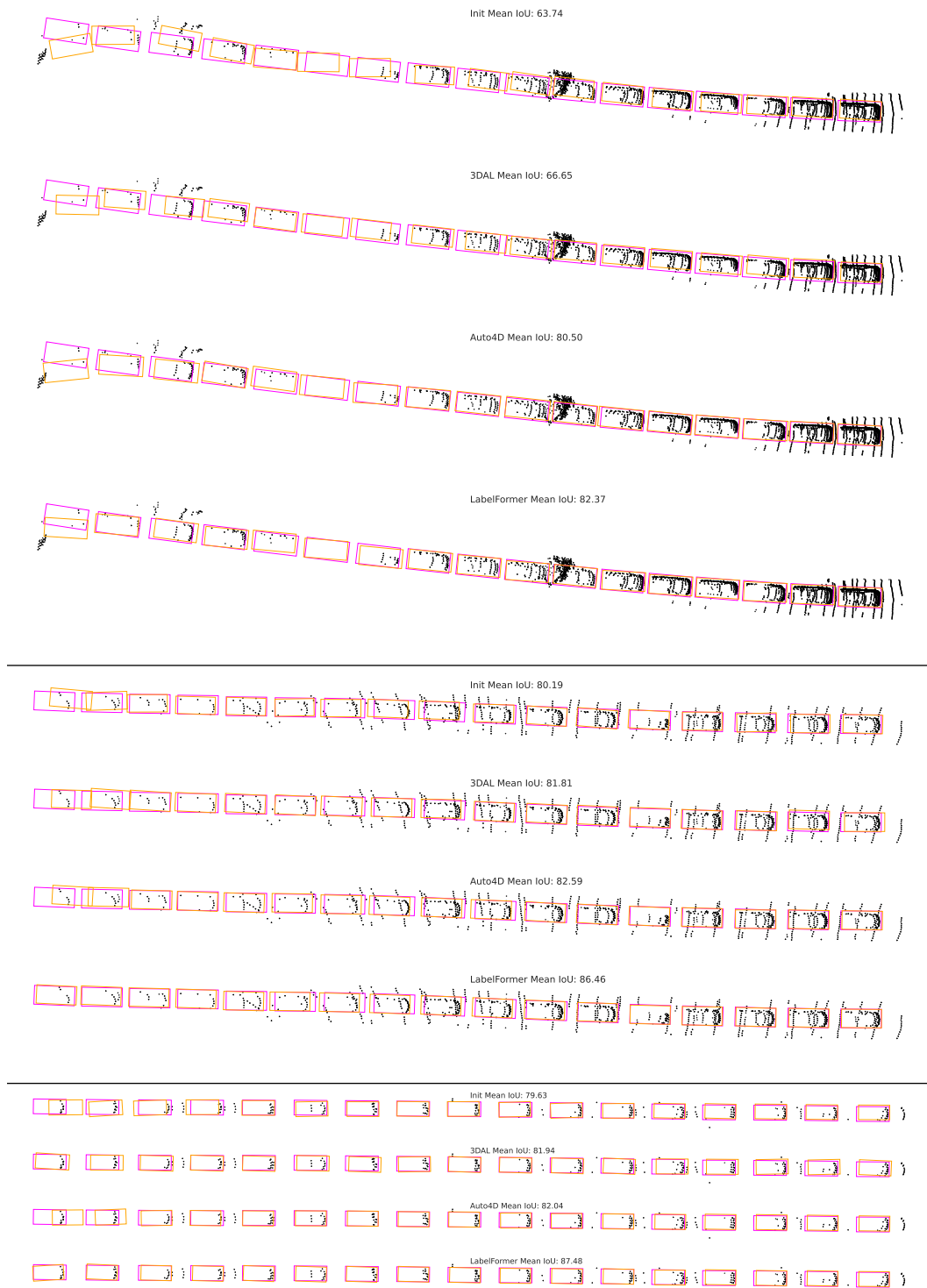


Figure 4: [Highway] More Qualitative results

References

- [1] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.
- [2] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. 2016.
- [3] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [4] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12697–12705, 2019.
- [5] Y. Chen, J. Liu, X. Zhang, X. Qi, and J. Jia. Voxelnext: Fully sparse voxelnet for 3d object detection and tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.
- [6] B. Yang, M. Bai, M. Liang, W. Zeng, and R. Urtasun. Auto4d: Learning to label 4d objects from sequential point clouds. *CoRR*, abs/2101.06586, 2021. URL <https://arxiv.org/abs/2101.06586>.
- [7] C. R. Qi, Y. Zhou, M. Najibi, P. Sun, K. Vo, B. Deng, and D. Anguelov. Offboard 3d object detection from point cloud sequences. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6134–6144, June 2021.
- [8] X. Weng, J. Wang, D. Held, and K. Kitani. 3D Multi-Object Tracking: A Baseline and New Evaluation Metrics. *IROS*, 2020.
- [9] O. Press, N. Smith, and M. Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*, 2022. URL <https://openreview.net/forum?id=R8sQPpGCv0>.
- [10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.