Point Cloud Forecasting as a Proxy for 4D Occupancy Forecasting

Tarasha Khurana* Peiyun Hu* David Held Deva Ramanan

Carnegie Mellon University

In this supplement, we extend our discussion of the proposed reformulation of point cloud forecasting into 4D occupancy forecasting. Specifically, we discuss details about the network architecture of the proposed approach in Sec. 1, further results on nuScenes, KITTI-Odometry and ArgoVerse2.0 in Sec. 2, and the quality of our forecasts separately on the foreground and background points in Sec. 3.

1. Network details

Architecture implementation We build on top of the encoder-decoder architecture first proposed by Zeng *et al*. [10] for neural motion planning. We extend the version of this architecture used by Khurana *et al*. [4] for forecasting occupancy in the birds'-eye-view. The only difference between our setup and that used in prior work [4], is that we treat our 4D voxel grid (X x Y x Z x T) as a reshaped 3D voxel grid (X x Y x ZT), where the Z or height dimension is incorporated into the channel dimension of the input, allowing us to still make use of 2D convolutions on a 4D voxel grid. This means that every channel in the input, represents a slice of the world through the height and time dimensions.

Differentiable renderer We extend the differentiable raycaster developed by Khurana *et al*. [4] to 3D and employ it as the differentiable voxel renderer in our approach. As in the prior work, we define our set of rays using the position of the egovehicle in the global coordinate frame as the origin, and all the LiDAR returns as the end points for the rays. The 4D voxel grid is initialized with three labels - empty, occupied and unknown based on the returns in the LiDAR sweeps. Each ray is traversed using a fast voxel traversal algorithm [2]. Given all the voxels and their occupancies along a ray, we compute the expected distance the ray travels through the voxel grid. This is same as volume rendering but in a discretized grid [6]. The gradient of the loss between this expected distance and the groundtruth distance is backpropagated to all the voxels traversed by the ray. Note that when a ray does not terminate within the voxel grid volume, we put all the probability mass of occupancy at the boundary of the voxel grid, similar to Mildenhall *et al*. [6]. This means that when a ray passes through occupancy regions that are empty (refer occupancy visuals in the main draft and supplementary video), the rays results in a point at the boundary of the voxel grid.

Dataset training and testing splits We use the official train and validation splits of nuScenes and ArgoVerse2.0. Only when comparing results on KITTI-Odometry with ST3DCNN [5], we follow their dataset splits for training and testing. These dataset splits allow us to draw apples-to-apples comparisons with state-of-the-art approaches.

2. Additional results

2.1. nuScenes

Results with confidence thresholding We supplement the results in the main paper by evaluating the point cloud forecasts of SPFNet [8] and S2Net [7] by thresholding points at a recommended confidence threshold of 0.05. Qualitatively in Fig. 1, we observe point clouds from SOTA that only consist of high confidence LiDAR returns close to the ground plane, because of which we perform quantitatively much better than these baselines on our ray-based metrics. We summarise these results in Tab. 1.

Access to ground-truth egoposes during evaluation Note that in our proposed formulation of 4D occupancy forecasting, we view the LiDAR point clouds used during training as just another observation of the world, which in our case, happens to come from the view of the ego-vehicle. In reality, this LiDAR measurement of occupancy could have also come from any other observer in the world. Similarly, during evaluation, the only LiDAR measurement we have access to comes from the view of the ego-vehicle, making this the only datapoint to evaluate our occupancy forecasts against. This creates an apparent advantage for our method when comparing to point cloud forecasting approaches because they do not have access to ground-truth

*Equal contribution

Method	Horizon	L1 (m)	AbsRel (%)	Chamfer Distance (m^2) Near-field	Vanilla
S2Net [7]	1s	2.88	20.57	4.61	11.77
	3s	4.97	24.79	13.10	30.95
SPFNet [8]	1s	5.30	30.12	21.24	45.12
	3s	5.70	28.65	20.99	44.71
Ray tracing	1s	1.50	14.73	0.54	**0.90**
	3s	2.44	26.86	1.66	**3.59**
Ours	1s	**1.40**	**10.37**	**1.41**	2.81
	3s	**1.71**	**13.48**	**1.40**	4.31

Table 1. Results on nuScenes [3] with confidence filtering on SPFNet and S2Net. As described in the main paper we threshold the points at a recommended confidence threshold of 0.05. We see that the conclusions made from the proposed metrics are more in line with the qualitative results in Fig. 1. This once again reiterates the need for metrics that intuitively evaluate the underlying *geometry* of the scene instead of uncorrelated samples of the scene (e.g., points in space).

Method	GT Egoposes	L1 (m)	AbsRel (%)	Chamfer Distance (m^2) Near-field	Vanilla
Ray tracing	Yes	2.44	26.86	1.66	3.59
Ray tracing	No	2.50	26.35	1.60	**3.39**
Ours	Yes	**1.71**	**13.48**	**1.40**	4.31
Ours	No	1.84	13.95	1.50	4.50

Table 2. We experiment with using a simple linear dynamics based motion planner that can replace the ground-truth future egoposes used in our analysis. Our experiments prove that even in the absence of access to ground-truth future egoposes – which are not a concern from the viewpoint of our formulation but only a means to evaluate the occupancy predictions – simple linear dynamics models such as those based on constant velocity, suffice.

egoposes from the future. To alleviate this concern, first, we use the future ground-truth egoposes to align all point cloud forecasts to a global coordinate frame. Only after doing this, all the reported metrics are computed for the baselines. Second, we employ a simple motion planner based on linear dynamics, and use these planned future egoposes for evaluating our own method. We see that the metrics drop marginally, showing that the dependence of our method on ground-truth egoposes from the future is not a concern. This is also true for the ray tracing baseline, results of which are summarised in Tab. 2.

2.2. KITTI-Odometry

Qualitative results We supplement the quantitative results in the main paper with qualitative results in Fig. 2. As noted before, the trends are similar to nuScenes.

2.3. ArgoVerse2.0

We benchmark ArgoVerse2.0 in Tab. 3 and compare the ray tracing baseline to our method. We see that the ray tracing baseline is strong and performs better than our method

Method	Horizon	L1 (m)	AbsRel (%)	Chamfer Distance (m^2) Near-field	Vanilla
Ray tracing	1s	2.39	15.43	**0.56**	**1.90**
	3s	3.72	25.24	2.50	**11.59**
Ours	1s	**2.25**	**10.25**	1.53	60.94
	3s	**2.86**	**14.62**	**2.20**	69.81

Table 3. Quantitative results on the ArgoVerse2.0 [9] dataset. We compare our method trained on the ArgoVerse2.0 dataset to the ray tracing baseline and find similar trends to nuScenes and KITTI-Odometry.

in terms of the Chamfer distance. Yet, our method is able to forecast better scene geometry in the near-field, than the ray tracing baseline as suggested by the ray-based metrics.

Note that the high vanilla Chamfer distance for ArgoVerse2.0 is due to the fact that the its LiDAR is long-range (up to 200m) and we focus on points only withing the bounded volume. We also clarify that we always test cross-sensor generalization between KITTI-Odometry and ArgoVerse2.0, with training on ArgoVerse2.0 because (1) both datasets have the same number of LiDAR beams and point clouds are captured at the same frequency, and (2) ArgoVerse2.0 is a much larger and diverse dataset than KITTI-Odometry that is suitable for pretraining.

3. Foreground vs. background query rays

In order to further analyse the variants of our architecture, we separate the query rays as belonging to foreground or background regions, using the labels from nuScenes' LiDARSeg [1]. We evaluate both the regions using both the new and old metrics in Table 5 and Table 6.

Poor performance on foreground objects Our main observation is that all the variants perform poorly on the foreground objects (which includes moving or stationary foreground objects) as compared to the background. This is because a large number of rays and voxels (more than 90%) belong to background regions and thus, the foreground objects are downweighted during the training process. Even when the combined evaluation of foreground and background regions is considered (Table 3 in main draft), we see that the poor performance on the foreground fails to materialize in the metrics. This hints are improving the metrics and methods to focus more on the forecasting of foreground objects, especially those in motion.

Strengths of each variant Another observation stemming from the above fact is that even with this disentangled evaluation on foreground, the *static* variant is the strongest baseline for short-horizon forecasting (1s). On 3s forecasting, the *dynamic* variant shines on the ray-based evaluation

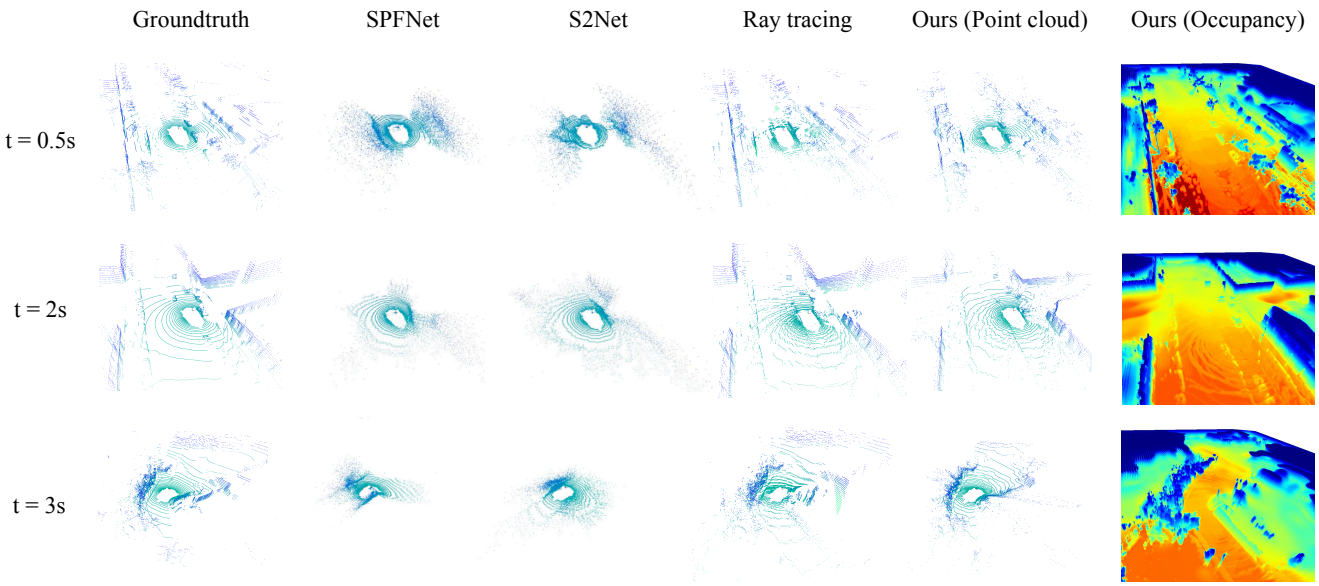

Figure 1. Qualitative results on the nuScenes dataset on three different sequences at different time horizons. We compare the point cloud forecasts of our approach with the aggregation-based ray tracing baseline and S2Net [7], SPFNet [8] with confidence filtering, after applying a recommended confidence threshold of 0.05 on the point clouds. Our forecasts look significantly crisper than the SOTA, however we see that the ray tracing baseline is also a strong baseline. We visualize a render of the learnt occupancy and the color encodes height along the z-axis.

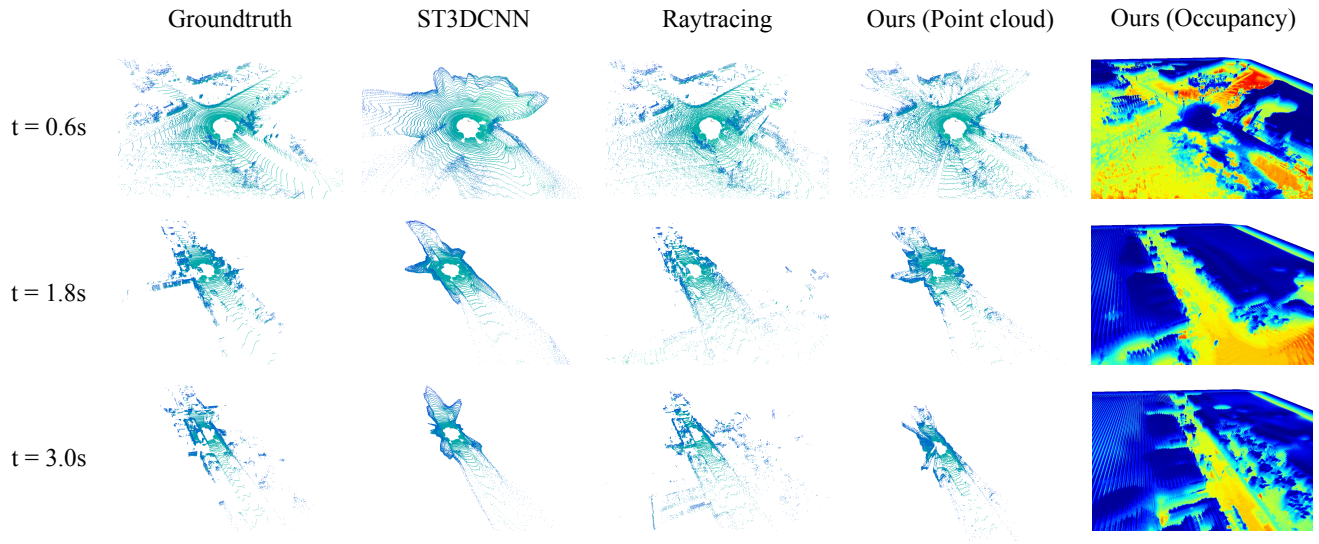

Figure 2. Qualitative results on KITTI-Odometry on three different sequences at different time horizons. We compare the point cloud forecasts of ST3DCNN [5] and the ray tracing baseline. We see that this SOTA is qualitatively more geometry-aware than the SOTA on nuScenes. However, our method is still more reflective of the true rigid geometry of the underlying world. We visualize a render of the learnt occupancy and the color encodes height along the z-axis.

Config	Horizon	Pedestrians				Vehicles				All foreground				Background			
		L1	AbsRel	Chamfer Dist. N.f.	Chamfer Dist. Vanilla	L1	AbsRel	Chamfer Dist. N.f.	Chamfer Dist. Vanilla	L1	AbsRel	Chamfer Dist. N.f.	Chamfer Dist. Vanilla	L1	AbsRel	Chamfer Dist. N.f.	Chamfer Dist. Vanilla
Ray tracing	1s	**6.09**	37.18	**61.30**	**66.79**	**3.53**	28.82	**16.92**	**21.19**	3.72	34.47	**16.09**	19.45	1.39	12.51	**0.49**	**0.85**
Ours	1s	6.43	**34.89**	79.63	68.60	3.61	**25.28**	21.47	22.59	**3.61**	**28.33**	18.19	**19.05**	**1.33**	**8.82**	1.44	3.02
Raytracing	3s	7.84	46.42	92.86	92.97	5.29	44.25	26.99	38.22	5.52	51.48	25.66	35.32	2.27	23.50	1.60	**3.48**
Ours	3s	**6.58**	**34.72**	**78.47**	**71.99**	**4.11**	**29.73**	**22.28**	**28.36**	**4.14**	**33.22**	**18.59**	**22.57**	**1.61**	**11.48**	**1.43**	4.63

Table 4. We extend our metrics analysis to pedestrians, vehicles, movable foreground and static background objects separately. Since, we are not able to compute these category-wise metrics for the state-of-the-art [7, 8] due to historical reasons, we do this for the ray tracing baseline. In summary, we find that our method outperforms this otherwise strong baseline at long-horizon forecasting. However, in the short-horizon the ray tracing baseline is a strong one; sometimes doing better and other times performing at par with our proposed method.

Arch.	Horizon	L1 (m)	AbsRel (%)	Chamfer Distance (m^2) Near-field	Chamfer Distance (m^2) Vanilla
S	1s	3.28	26.14	13.52	15.35
	3s	4.10	32.25	19.25	23.05
D	1s	3.61	28.33	18.19	19.05
	3s	4.14	33.22	18.59	22.57
S + R	1s	3.28	25.34	13.95	15.01
	3s	4.11	31.65	19.91	25.29

Table 5. Performance analysis on **foreground** query rays on nuScenes [3] for the different architecture variants introduced in the main draft, static (S), dynamic (D) and residual (S+R).

Arch.	Horizon	L1 (m)	AbsRel (%)	Chamfer Distance (m^2) Near-field	Chamfer Distance (m^2) Vanilla
S	1s	1.22	7.89	1.10	3.74
	3s	1.64	11.65	1.43	4.00
D	1s	1.33	8.82	1.44	3.02
	3s	1.61	11.48	1.43	4.63
S + R	1s	1.29	8.48	1.07	3.52
	3s	1.74	12.01	1.56	3.78

Table 6. Performance analysis on **background** query rays on nuScenes [3] for the different architecture variants introduced in the main draft, static (S), dynamic (D) and residual (S+R).

of background objects (some unseen background regions may only appear at future timesteps) and the *residual* variant shines on the ray-based evaluation of foreground objects (possibly decouples the foreground from background regions better).

Comparison to the ray tracing baseline Given the strength of the ray tracing baseline, we investigate its performance on foreground and background objects in comparison to our approach in Tab. 4. This time we further divide foreground objects into subcategories of pedestrains and vehicles, while also reporting the metrics on all foreground objects. Note that the according to the vocabulary of nuScenes, apart from different types of pedestrians and vehicles, miscellaneous movable objects like traffic cones and barriers are included in the umbrella category of foreground objects. We have the following findings:

1. For long-horizon forecasting, our method consistently does better than the ray tracing baseline, for both all types of foreground objects and background objects.

2. For short-horizon forecasting, the ray tracing baseline performs at par with our method and sometimes even better (on most types of foreground objects), hence proving to be a strong yet simple and non-learnable approach.

References

[1] nuScenes LiDARSeg. https://www.nuscenes.org/nuscenes#lidarseg. 2

[2] John Amanatides and Andrew Woo. A Fast Voxel Traversal Algorithm for Ray Tracing. In *EG 1987-Technical Papers*. Eurographics Association, 1987. 1

[3] 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. *arXiv preprint arXiv:1903.11027*, 2019. 2, 4

[4] Tarasha Khurana, Peiyun Hu, Achal Dave, Jason Ziglar, David Held, and Deva Ramanan. Differentiable raycasting for self-supervised occupancy forecasting. In *European Conference on Computer Vision*, pages 353–369. Springer, 2022. 1

[5] Benedikt Mersch, Xieyuanli Chen, Jens Behley, and Cyrill Stachniss. Self-supervised point cloud prediction using 3d spatio-temporal convolutional networks. In *Conference on Robot Learning*, pages 1444–1454. PMLR, 2022. 1, 3

[6] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer, 2020. 1

[7] Xinshuo Weng, Junyu Nan, Kuan-Hui Lee, Rowan McAllister, Adrien Gaidon, Nicholas Rhinehart, and Kris M Kitani. S2net: Stochastic sequential pointcloud forecasting. In *European Conference on Computer Vision*, pages 549–564. Springer, 2022. 1, 2, 3, 4

[8] Xinshuo Weng, Jianren Wang, Sergey Levine, Kris Kitani, and Nicholas Rhinehart. Inverting the pose forecasting pipeline with spf2: Sequential pointcloud forecasting for sequential pose forecasting. In *Proceedings of the 2020 Conference on Robot Learning*, pages 11–20, 2021. 1, 2, 3, 4

[9] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, et al. Argoverse 2.0: Next generation datasets for self-driving perception and forecasting. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. 2

[10] Wenyuan Zeng, Wenjie Luo, Simon Suo, Abbas Sadat, Bin Yang, Sergio Casas, and Raquel Urtasun. End-to-end interpretable neural motion planner. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8660–8669, 2019. 1