Differentiable Raycasting for Self-supervised Occupancy Forecasting

Supplementary Material

In this supplement, we discuss more details of our experimental setup in Sec. A, discuss supplementary evaluation of occupancy forecasting in Sec. B and analyse the quantitative and qualitative performance of our motion planning architecture further in Sec. C.

A Experimental Setup

A.1 Network Architecture

Architecture Implementation We use the same neural network architecture as proposed by Zeng *et al.* [6] and developed on by Hu *et al.* [4]. Different from these two networks, we use two decoders, one that predicts the emergent occupancy cost maps, and one that predicts the residual cost maps. The differentiable raycaster proposed by us acts as a layer over the occupancy cost maps, that produces raycast sweeps for 7 future timesteps (accounting for three seconds in the future).

Freespace is computed from these sweeps and it is used in 3 places in the network: (1) in computing a dense per-pixel classification loss with the groundtruth freespace, (2) in computing the final cost maps which are a sum of the freespace and the residual cost maps, and (3) in computation of the cost margin for the planning loss.

Input and output We follow the same input and output BEV data format as that used in Hu *et al.* [4] for nuScenes, except that we now take input from [-2s, 0s]. For nuScenes, this means 20 input timestamps and a stack of $704 \times 400 \times 20$ size. For ONCE, since the LiDAR sweeps are collected at 2Hz, this stack is of size $704 \times 400 \times 5$ to accomodate 2s of input data. The output of the network for both the datasets is of size $704 \times 400 \times 7$ to account for 3s of forecasts at a 0.5s interval, starting at the 0th timestep. Each pixel in the BEV map covers an area of $0.2m \times 0.2m$. To compute groundtruth freespace cost maps, we apply ground segmentation [4,3] to the output LiDAR sweep and raycast as described in the main paper.

Differentiable raycaster First, we collect a set of rays, with origin as the position of the ego-vehicle in the world coordinate frame and endpoints as the endpoints in a groundtruth LiDAR sweep. For a given ray (origin and direction), we find the voxels that the ray travels in the BEV LiDAR scan using a fast voxel traversal algorithm proposed by Amanatides *et al.*[1]. Given all the voxels along a ray, we perform a soft raycast along the ray as follows: we sample occupancy states given the predicted occupancy probabilities, raycast to get free vs occluded space, and average the raycasts over all samples. In practice, we do this analytically by computing the expectation, as done by many prior works based on volume rendering [5].

A.2 Data-driven sampler

Following prior work that uses data-driven trajectory sampling techniques for evaluating the performance in mapless driving scenarios [2], we curate a dataset of expert trajectories by binning the trajectories in the train set by their velocity. Once this dataset is curated during preprocessing, we use retrieval based on the past timestep's speed and direction profiles to index into the appropriate bin in the dataset. When a velocity is not available in the set of data-driven trajectories, we compute the nearest speed and angle from the set, for a given sample. From this nearest bin, we randomly sample 200 valid trajectories and append them to our set of 2000 model-based trajectories.

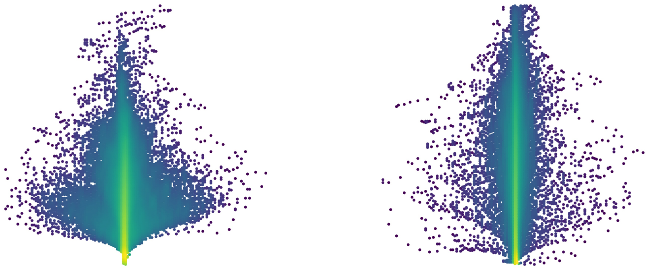

Fig. 1. Distribution of train trajectories in nuScenes (**left**) and ONCE (**right**).

This approach avoids arbitrary choice of steering profiles for the ONCE dataset, since this information in unknown in ONCE (note that this is available for nuScenes with the CAN bus data). This is useful because in comparison to nuScenes, the ONCE dataset is composed of a complementary set of trajectories, as shown in Fig. 1. Using this data-driven trajectory sampler in conjunction with the standard trajectory sampler gives us a complete coverage of possible future trajectories, including the ones that appear the most in the ONCE dataset.

B Occupancy Forecasting

We supplement the evaluation of occupancy forecasting on ONCE and nuScenes in the main paper by providing complete results of Fig. 7 in Tab. 1. The unlabeled

Dataset	Size	$\frac{\lvert \mathbf{d}-\hat{\mathbf{d}} \rvert}{\mathbf{d}}$ (↓)	BCE (↓)	F1 (↑)	AP (↑)
ONCE	2,000	0.336	0.109	0.649	0.776
	4,000	0.260	0.102	0.711	0.814
	8,000	**0.243**	**0.097**	**0.787**	**0.827**
ONCE (unlabeled)	2,000	0.598	0.246	0.376	0.493
	4,000	0.589	0.236	0.384	0.502
	8,000	0.553	0.200	0.460	0.553
	22,000	0.536	0.200	0.466	0.576
	86,000	**0.513**	**0.174**	**0.495**	**0.607**
nuScenes	2,000	0.299	0.184	0.726	0.804
	4,000	0.280	0.169	0.749	0.826
	8,000	0.261	0.157	0.761	0.843
	16,000	0.244	0.148	0.774	0.859
	22,000	**0.242**	**0.140**	**0.777**	**0.863**

Table 1. Supplementary table for the evaluation of occupancy forecasting on ONCE-val and nuScenes-val with models trained on different subsets of the ONCE labeled, unlabeled and nuScenes train set.

subsets of ONCE do not include samples from the labeled train set. As described, increasing the amount of training data directly impacts the improvement in performance. It is worth noting the performance difference between the 8k set of ONCE-labeled and ONCE-unlabeled. The higher metrics on the labeled set indicates that the ONCE labeled set is much higher quality and falls in the same data distribution as compared to the val set. nuScenes training subsets also show increasing performance with increase in data.

C Motion Planning

C.1 Planning on ONCE

Quantitative Analysis This section supplements our results on the ONCE dataset from the main paper. We show the complete results of Fig. 12 in Tab. 4. Note that as the amount of data is increased during training, the L2 error and box collision rate decreases dramatically. Even though box collision rate is a stricter metric than point collision rate, we see a consistent trend in it at the longest horizon. Our best model beats the neural motion planner [6] baseline described in the main paper. Note that such a baseline can only be trained with the labeled training set of ONCE (with 8K samples), whereas all the raw unlabelled LiDAR logs in ONCE can be used by our method since it is self-supervised.

We also conduct an ablative study of our approach on the ONCE dataset in Tab. 3. Note that since the hyperparameters are not tuned for the ONCE dataset, the best performing method on the Box Collision metric at 3s horizon is by Hu *et al.* [4]. Intuitively, this difference in performance shows that the trajectories

Training	L2 Error (m)			Point Collision (%)			Box Collision (%)		
size	1s	2s	3s	1s	2s	3s	1s	2s	3s
8,000	0.84	2.26	4.45	0.00	0.04	1.06	0.04	0.14	2.54
2,000	1.97	4.37	8.18	0.07	0.39	1.84	0.77	2.36	5.05
4,000	1.13	2.79	5.33	0.00	0.04	1.02	0.14	0.81	3.43
8,000	1.00	2.33	4.43	0.00	0.04	**0.74**	0.04	**0.28**	2.79
22,000	0.71	1.87	3.62	0.00	0.14	1.06	0.04	0.39	2.65
94,000	**0.56**	**1.49**	**2.90**	**0.00**	**0.04**	0.99	**0.00**	0.39	**2.47**

Table 2. Planning metrics at different amount of training data on ONCE-val. First row corresponds to our reimplementation of the neural motion planner [6] baseline described in the main paper.

	Cost	Mid	Diff.	L2 Distance (m)			Point Collision (%)			Box Collision (%)		
	Margin	Task	Raycast	1s	2s	3s	1s	2s	3s	1s	2s	3s
(a)	-	-	-	**0.61**	**1.64**	**3.33**	**0.00**	**0.00**	1.02	**0.00**	0.42	2.47
(b)	✓	-	-	0.80	2.12	4.15	**0.00**	**0.00**	0.78	**0.00**	**0.18**	**1.84**
(c)	-	✓	-	0.89	2.40	4.78	**0.00**	0.04	1.63	**0.00**	0.35	3.85
(d)	✓	✓	-	0.90	2.49	4.99	**0.00**	0.04	1.10	0.07	0.25	2.61
(e)	✓	✓	✓	1.00	2.33	4.43	**0.00**	0.04	**0.74**	0.04	0.28	2.79

Table 3. Ablation studies on ONCE. Note that (a) is IL, (b) is FF, and (e) is **Ours**.

selected by our planner pass close to the objects in the environment, such that they incur a box collision but not point collision. This is expected as even though we outperform Hu *et al.* [4] on occupancy forecasting, the guided planning loss used optimizes for point collision by summing per way-point occupancy cost instead of box collision, on which we outperform all other methods at 3s horizon.

Qualitative Analysis (Videos) We show a qualitative result of our approach in `result.mp4` which is a video version of one of the qualitative results in the main paper. The video first shows the input to our network, followed by a visualization of the predicted future **emergent/latent occupancy** and the **sweep raycast** differentiably from this occupancy (referred to as $\hat{d}_2$ in Eq. 5 in the main paper). Finally, we show the **total cost maps**, which are a sum of the occupancy and residual cost maps and the **final output trajectory**.

Additionally, we show two examples of the evolution of predicted emergent occupancy. In the first visual in `evolution1.mp4`, note how a straight moving car's possible future locations evolve with time. Similarly for a car turning right at an intersection in `evolution2.mp4`, notice how the multiple possible futures of the car grow into a triangular blob, indicating that according to the occupancy probability, the car could have moved at any angle, either moving straight across the intersection or turning right.

Dataset	Training size	L2 Error (m)			Point Collision (%)			Box Collision (%)		
		$1s$	$2s$	$3s$	$1s$	$2s$	$3s$	$1s$	$2s$	$3s$
nuScenes	-	0.76	1.61	3.23	0.00	0.00	0.15	0.04	0.15	0.98
ONCE	2,000	2.10	4.39	7.74	**0.00**	0.25	1.45	0.32	1.94	4.80
	4,000	1.09	2.73	5.15	**0.00**	0.04	0.99	0.07	0.53	3.28
	8,000	0.87	2.24	4.32	**0.00**	**0.00**	0.78	0.04	0.25	2.37
	22,000	0.71	1.92	3.74	**0.00**	0.28	1.02	0.07	0.67	2.65
	94,000	**0.50**	**1.32**	**2.61**	**0.00**	0.04	**0.71**	**0.00**	**0.21**	**1.94**

Table 4. Evaluation of planning metrics on nuScenes and ONCE by adding the occupancy cost maps to residual cost maps during training.

C.2 Ablations on training architecture

While we compute the predicted cost margin in the main paper by summing the egocentric-freespace cost maps with the residual cost maps for an apples-to-apples comparison with Hu *et al.*[4], a more natural training architecture for motion planning would sum up the occupancy and residual cost maps during training, similar to the test-time architecture. Such an architecture would compute egocentric-freespace only for self-supervision with the multi-task loss and use the occupancy cost maps for motion planning. In Tab. 4, we evaluate training with this ablated architecture. Note that since the occupancy cost maps are now optimized directly during training, the performance across all metrics increases.

C.3 Limitations

We highlight a few limitations of our work. First, our self-supervised emergent occupancy does not offer semantics (e.g., traffic light and lane information) that is crucial for urban navigation. Despite this, we show that learning to drive with future occupancy is a safe fallback option in industrial autonomous driving. Second, BEV occupancy by itself does not handle overhead structures (e.g., trees, overpass); this may be mitigated by learning which occupied voxels are 'passable' during differentiable raycasting. Third, we rely on open-loop evaluation, where the world (incorrectly) unfolds in the same manner as the expert trajectory. Although this can be corrected in a closed-loop setup, with our work we show that optimizing for collision metrics, with or without L2 error, can act as a proxy for learning to drive safe in real-world. Fourth, our method assumes accurate ego-motion during training but does not require it at test time. Finally, as the supplementary video highlights, our occupancy estimates diffuse over time, capturing multiple futures. However, we posit that future occupancy can be made more robust by constraining it with scene flow.

References

1. Amanatides, J., Woo, A.: A Fast Voxel Traversal Algorithm for Ray Tracing. In: EG 1987-Technical Papers. Eurographics Association (1987). https://doi.org/10.2312/egtp.19871000 1
2. Casas, S., Sadat, A., Urtasun, R.: Mp3: A unified model to map, perceive, predict and plan. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14403–14412 (2021) 2
3. Himmelsbach, M., Hundelshausen, F.V., Wuensche, H.J.: Fast segmentation of 3d point clouds for ground vehicles. In: Intelligent Vehicles Symposium (IV), 2010 IEEE. pp. 560–565. IEEE (2010) 1
4. Hu, P., Huang, A., Dolan, J., Held, D., Ramanan, D.: Safe local motion planning with self-supervised freespace forecasting. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12732–12741 (2021) 1, 3, 4, 5
5. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: European conference on computer vision. pp. 405–421. Springer (2020) 2
6. Zeng, W., Luo, W., Suo, S., Sadat, A., Yang, B., Casas, S., Urtasun, R.: End-to-end interpretable neural motion planner. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8660–8669 (2019) 1, 3, 4