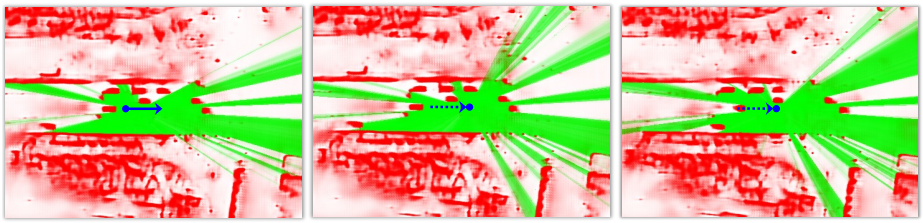
Differentiable Raycasting for Self-supervised Occupancy Forecasting

Tarasha Khurana[1⋆], Peiyun Hu[2∗], Achal Dave[3], Jason Ziglar[2], David Held[1,2], and Deva Ramanan[1,2]

[1] Carnegie Mellon University
[2] Argo AI
[3] Amazon



(a) Sensor position y, Scene s (b) New sensor position $y + \Delta y$, Scene s (c) New sensor position $y + \Delta y$, New scene $s + \Delta s$

Fig. 1. We propose **emergent occupancy** as a novel self-supervised representation for motion planning. Occupancy is independent of changes in sensor pose Δy, which is in contrast to prior work on self-supervised learning from LiDAR [28,29,15,9], specifically, **ego-centric freespace** [9], which changes with (**a-b**) sensor pose motion Δy and (**b-c**) scene motion Δs. We use differentiable raycasting to naturally decouple ego motion from scene motion, allowing us to learn to forecast occupancy by self-supervision from pose-aligned LiDAR sweeps.

Abstract. Motion planning for safe autonomous driving requires learning how the environment around an ego-vehicle evolves with time. Ego-centric perception of driveable regions in a scene not only changes with the motion of actors in the environment, but also with the movement of the ego-vehicle itself. Self-supervised representations proposed for large-scale planning, such as ego-centric freespace, confound these two motions, making the representation difficult to use for downstream motion planners. In this paper, we use *geometric occupancy* as a natural alternative to view-dependent representations such as freespace. Occupancy maps naturally disentagle the motion of the environment from the motion of the ego-vehicle. However, one cannot directly observe the full 3D occupancy of a scene (due to occlusion), making it difficult to use as a signal for learning. Our key insight is to use *differentiable raycasting* to "render" future occupancy predictions into future LiDAR sweep predictions, which can be compared with ground-truth sweeps for self-supervised learning. The use of differentiable raycasting allows occupancy to *emerge* as an internal representation within the forecasting network. In the absence of groundtruth occupancy, we quantitatively evaluate the forecasting of raycasted LiDAR sweeps and show improvements of upto 15 F1 points.

⋆ equal contribution

For downstream motion planners, where emergent occupancy can be directly used to guide non-driveable regions, this representation relatively reduces the number of collisions with objects by up to 17% as compared to freespace-centric motion planners.

1 Introduction

To navigate in complex and dynamic environments such as urban cores, autonomous vehicles need to perceive actors and predict their future movements. Such knowledge is often represented in some form of forecasted occupancy [23], which downstream motion planners rely on to produce safe trajectories. When tackling the tasks of perception and prediction, standard solutions consist of perceptual modules such as object detection, tracking, and trajectory forecasting, which require a massive amount of object track labels. Such solutions do not scale given the speed that log data is being collected by large fleets.

Freespace versus occupancy: To avoid the need for costly human annotations, and to enable learning at scale, self-supervised representations such as ego-centric freespace [9] have been proposed. However, such a representation couples the motion of the world with the motion of the ego-vehicle (Fig. 1). Our key innovation in this paper is to learn an ego-pose independent and explainable representation for safe motion planning, which we call *emergent occupancy*. Emergent occupancy decouples ego motion and scene motion using differentiable raycasting: we design a network that learns to "space-time complete" the future volumetric state of the world (in a world-coordinate frame) given past LiDAR observations. Consider an ego-vehicle that moves in a static scene. Here, LiDAR returns (even when aligned to a world-coordinate frame) will still *swim* along the surfaces of the fixed scene (Fig. 2). This implies that even when the world is static, most of what the ego-vehicle observes through the LiDAR sensor appears to move with complex nonlinear motion, but in fact those observations can be fully explained by static geometry and ego-motion (via raycasting). LiDAR forecasters need to implicitly predict this ego-motion of the car to produce accurate future returns. However, we argue that such prediction doesn't make sense for autonomous agents that *plan* their future motion. Importantly, our differentiable raycasting network has access to future camera ego-poses as *input*, both during training (since they are available in archival logs) and testing (since state-of-the-art planners explicitly search over candidate trajectories).

Self-supervision: Note that ground-truth future volumetric occupancy is largely unavailable without human supervision, because the full 3D world is rarely observed; the ego-vehicle only sees a limited number of future views as recorded in a single archival log. To this end, we apply a differentiable raycaster that projects the forecasted volumetric occupancy into a LiDAR sweep, as seen by the future ego-vehicle motion in the log. We then use the difference between the raycasted sweep and actual sweep as a signal for self-supervised learning, allowing us to train models on massive amounts of unannotated logs.

Planning: Lastly, we show that such forecasted space-time occupancy can be jointly learned with space-time costmaps for end-to-end motion planning.

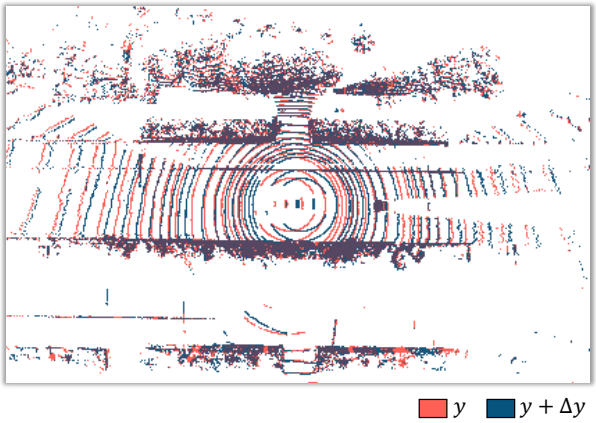


Fig. 2. We pose-align two succesive LiDAR sweeps of a static scene s to a common world coordinate-frame (using the notation of Fig. 1). Even though there is zero scene motion Δs, points appear to drift or *swim* across surfaces. This is due to the fact that points are obtained by intersecting rays from a moving sensor Δy with static scene geometry. This in turn implies that points can appear to move since they are not tied to physical locations on a surface. This apparent movement ($\Delta \tilde{s}$) is in general a complex nonlinear transformation, even when the sensor motion Δy is a simple translation (as shown above). Traditional methods for self-supervised LiDAR forecasting [28,29,15,9] require predicting the complex transformation $\Delta \tilde{s}$ which depends on the unknown Δy, while our differentiable-raycasting framework assumes Δy is an *input*, dramatically simplifying the task of the forecasting network. From a planning perspective, we argue that the future (planned) change-in-pose *should* be an input rather than an output.

Owing to LiDAR self-supervision, we are able to train on recent unsupervised LiDAR datasets [13] that are orders of magnitude larger than their annotated counterparts, resulting in significant improvement in accuracy for both forecasted occupancy and motion plans. Interestingly, as we increase the amount of archival training data at the cost of zero additional human annotation, object shape, tracks, and multiple futures "emerge" in the arbitrary quantities predicted by our model despite there being no direct supervision on ground-truth occupancy.

2 Related Work

Occupancy as a scene representation: Knowledge regarding what is around an autonomous vehicle (AV) and what will happen next is captured in different representations throughout the standard modular perception and prediction (P&P) pipeline [11,26,4,24]. Instead of separate optimization of these modules [25,16], Sadat et al. [23] propose bird's-eye view (BEV) *semantic occupancy* that is end-to-end optimizable. As an alternative to *semantic occupancy*, Hu et al. [10] propose BEV *ego-centric freespace* that can be self-supervised by raycasting on aligned LiDAR sweeps. However, the ego-centric freespace entangles

motion from other actors, which is arguably more relevant for motion planning, with ego-motion. In this paper, we propose *emergent occupancy* to isolate motion of other actors. While we focus on self-supervised learning at scale, we acknowledge that for motion planning, some semantic labelling is required (e.g., state of a traffic light) which can be incorporated via semi-supervised learning.

Differentiable raycasting: Differentiable raycasting has shown great promise in learning the underlying scene structure given samples of observations for downstream novel view synthesis [14], pose estimation [30], etc. In contrast, our application is best described as "space-time scene completion", where we learn a network to predict an explicit space-time occupancy volume. Furthermore, our approach differs from existing approaches in the following ways. We use LiDAR sequences as input and raycast LiDAR sweeps given future occupancy and sensor pose. We work with explicit volumetric representations [12] for dynamic scenes with a feed-forward network instead of test-time optimization [18].

Self-supervision: Standard P&P solutions do not scale given how fast log data is collected by large fleets and how slow it is to curate object track labels. To enable learning on massive amount of unlabeled logs, supervision from simulation [8,5,6,7], auto labeling using multi-view constraints [20], and self-supervision have been proposed. Notably, tasks that can be naturally self-supervised by LiDAR sweeps e.g., scene flow [15] have the potential to generalize better as they can leverage more data. More recently, LiDAR self-supervision has been explored in the context of point cloud forecasting [27,28,29]. However, when predicting future sweeps given the history, as stated before, past approaches often tend to couple motion of the world with the motion of the ego-vehicle [27].

Motion Planning: An understanding of what is around an AV and what will happen next [25] is crucial. This is typically done in the bird's eye-view (BEV) space by building a modular P&P pipeline. Although BEV motion planning does not precisely reflect planning in the 3D world, it is widely used as the highest-resolution and computation- and memory-efficient representation [31,23,3]. However, training such modules often requires a massive amount of data. End-to-end learned planners requiring less human annotation have emerged, with end-to-end imitation learning (IL) methods showing particular promise [6,22,5]. Such methods often learn a neural network to map sensor data to either action (known as behavior cloning) or "action-ready" cost function (known as inverse optimal control) [17]. However, they are often criticized for lack of explainable intermediate representations, making them less accountable for safety-critical applications [19]. More recently, end-to-end learned but modular methods producing explainable representations, e.g., neural motion planners [31,23,3] have been proposed. However, these still require costly object track labels. Unlike them, our approach learns explainable intermediate representations that are explainable quantities for safety-critical motion planning without the need of track labels.

3 Method

Autonomous fleets provide an abundance of *aligned* sequences of LiDAR sweeps $\mathbf{x}$ and ego vehicle trajectories $\mathbf{y}$. How can we make use of such data to improve perception, prediction, and planning? In the sections to follow, we first define occupancy. Then we describe a self-supervised approach to predicting future occupancy. Finally, we describe an approach for integrating this forecasted occupancy into neural motion planners. Note that in the text that follows, we use ego-centric freespace and freespace interchangeably.

3.1 Occupancy

We define occupancy as the state of occupied space at a particular time instance. We use $\mathbf{z}$ to denote the true occupancy, which may not be directly observable due to visibility constraints. Let us write

$$\mathbf{z}[\mathbf{u}] \in \{0, 1\}, \mathbf{u} = (x, y, t), \mathbf{u} \in \mathbf{U} \tag{1}$$

to denote the occupancy of a voxel $\mathbf{u}$ in the space-time voxel grid $\mathbf{U}$, which can be *occupied* (1) or *free* (0). The spatial index of $\mathbf{u}$, i.e., (x, y) represents the spatial location from a bird's-eye view. Given a sequence of *aligned* sensor data and ego-vehicle trajectory $(\mathbf{x}, \mathbf{y})$, there may be multiple plausible occupancy states $\mathbf{z}$ that "explain" the sensor measurements. We denote this set of plausible occupancy states as $\mathbf{Z}$.

Forecasting Occupancy. Suppose we split an aligned sequence of LiDAR sweeps and ego-vehicle trajectory $(\mathbf{x}, \mathbf{y})$ into a historic pair $(\mathbf{x_1}, \mathbf{y_1})$ and a future pair $(\mathbf{x_2}, \mathbf{y_2})$. Our goal is to learn a function f that takes historical observations $(\mathbf{x_1}, \mathbf{y_1})$ as input and predicts emergent future occupancy $\hat{\mathbf{z}}_\mathbf{2}$. Formally,

$$\hat{\mathbf{z}}_\mathbf{2} = f(\mathbf{x_1}, \mathbf{y_1}), \tag{2}$$

If the true occupancy $\mathbf{z_2}$ were observable, we could directly supervise our forecaster, f. Unfortunately, in practice, we only observe LiDAR sweeps, $\mathbf{x}$. We show in the next section how to supervise f with LiDAR sweeps using differentiable raycasting techniques.

3.2 Raycasting

Given an occupancy estimate $\hat{\mathbf{z}}$, sensor origin $\mathbf{y}$ and directional unit vectors for rays $\mathbf{r}$, a differentiable raycaster $\mathcal{R}$ can raycast LiDAR sweeps $\hat{\mathbf{x}}$. We use $\hat{\mathbf{d}}$ to represent the expected distance these rays travel before hitting obstacles: $\hat{\mathbf{d}} = \mathcal{R}(\mathbf{r}; \hat{\mathbf{z}}, \mathbf{y})$. Then we can reconstruct the raycast LiDAR sweep $\hat{\mathbf{x}}$ as $\hat{\mathbf{x}} = \mathbf{y} + \hat{\mathbf{d}} * \mathbf{r}$.

3.3 Learning to Forecast Occupancy

Given the predicted occupancy $\hat{\mathbf{z}}_\mathbf{2}$ (Eq. 2), and the captured sensor pose $\mathbf{y_2}$, a differentiable raycaster $\mathcal{R}$ can take rays $\mathbf{r_2}$ as input and produce $\hat{\mathbf{d}}_\mathbf{2} = \mathcal{R}(\mathbf{r_2}; \hat{\mathbf{z}}_\mathbf{2}, \mathbf{y_2})$.

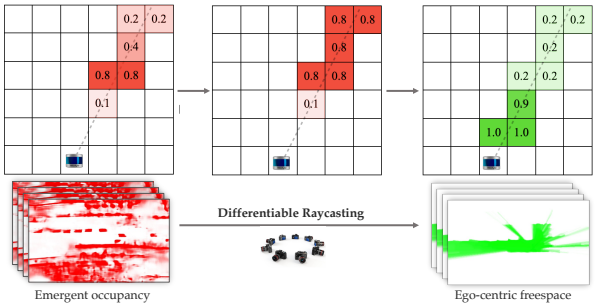


Fig. 3. Differentiable procedure for estimating ego-centric freespace from volumetric occupancy, necessary for computing the loss from (3). The left image depicts predicted emergent occupancy, on which we perform a cumulative max along the LiDAR ray from known sensor poses (middle), which is differentiable because it is essentially re-indexing. The result is then inverted to produce (soft) visible ego-centric freespace estimates. To identify BEV pixels along the LiDAR ray, we perform fast voxel traversal in 2D [1].

Note that this formulation allows us to decouple the motion of the world captured by change in occupancy, $\hat{\mathbf{z}}_{\mathbf{2}}$, and the motion of the ego-vehicle captured by change in sensor origin, $\mathbf{y}_{\mathbf{2}}$.

This also allows us to supervise $\hat{\mathbf{z}}_{\mathbf{2}}$ using a loss function that measures the difference between the raycast distance $\hat{\mathbf{d}}_{\mathbf{2}}$ and the ground-truth distance $\mathbf{d}_{\mathbf{2}}$.

$$L_r = \text{loss}(\hat{\mathbf{d}}_{\mathbf{2}}, \mathbf{d}_{\mathbf{2}}) \tag{3}$$

Loss function: One natural loss function might be distance between the raycast depth and measured depth along each ray. In practice, we care most about disagreements of freespace which can inform safe motion plans. To emphasize such disagreements, we define voxels encountered along the ray as having a free versus not-free binary label, and use a binary cross-entropy loss (summed over all voxels encountered by each ray until the boundary of voxel grid, ref. Fig. 3). We adopt an encoder-decoder architecture that predicts future emergent occupancy given historical LiDAR sweeps, differentiably raycasts future LiDAR sweeps and self-supervises using archival sweeps (ref. highlighted branch of Fig. 4 (a)).

3.4 Learning to Plan

The previous section described an approach for predicting future LiDAR returns via differentiable raycasting of BEV space-time occupancy maps. We now show that such costmaps can be integrated directly into an end-to-end motion planner that makes use of space-time costmaps for scoring candidate trajectories. We follow [9], but modify their derivation to take into account emergent occupancy.

Max-margin planning: We learn a model g to predict a space-time cost map, $\mathbf{c}_{\mathbf{2}}$, over future timestamps given past observations $(\mathbf{x}_{\mathbf{1}}, \mathbf{y}_{\mathbf{1}})$:

$$\mathbf{c}_{\mathbf{2}} = g(\mathbf{x}_{\mathbf{1}}, \mathbf{y}_{\mathbf{1}}), \text{ where } \mathbf{c}_{\mathbf{2}}[\mathbf{u}] \in \mathbb{R}, \mathbf{u} \in \mathbf{U}_{\mathbf{2}} \tag{4}$$

where $\mathbf{U_2}$ represents the space-time voxel grid over future timestamps. We define the cost of a trajectory as the sum of costs at its space-time way-points. The best candidate future trajectory according to the cost map is the one with the lowest cost:

$$\hat{\mathbf{y}}_\mathbf{2}^* = \arg\min_{\hat{\mathbf{y}} \in \mathbf{Y_2}} C(\hat{\mathbf{y}}; \mathbf{c_2}) = \arg\min_{\hat{\mathbf{y}} \in \mathbf{Y_2}} \sum_{\mathbf{u} \in \hat{\mathbf{y}}} \mathbf{c_2}[\mathbf{u}] \tag{5}$$

where $\mathbf{Y_2}$ represents the set of viable future trajectories.

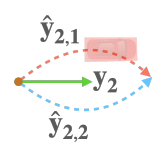
Loss function: We use a max-margin loss function, where the target cost of a candidate trajectory ($\hat{\mathbf{y}}$) is equal to the cost of the expert trajectory ($\mathbf{y_2}$) plus a margin. We can write the objective as follows:

$$L_p = \left[C\left(\mathbf{y_2}; \mathbf{c_2}\right) - \left(\min_{\hat{\mathbf{y}} \in \mathbf{Y_2}} C\left(\hat{\mathbf{y}}; \mathbf{c_2}\right) - D\left(\hat{\mathbf{y}}, \mathbf{y_2}\right) \right) \right]_+ \tag{6}$$

where $[\cdot]_+ = \max(\cdot, 0)$ and D is a function that quantifies the desired margin between the cost of a candidate trajectory and the cost of an expert trajectory. A common choice for D is Euclidean distance between pairs of way-points:

$$D(\hat{\mathbf{y}}_\mathbf{2}, \mathbf{y_2}) = ||\hat{\mathbf{y}}_\mathbf{2}, \mathbf{y_2}||_2 \tag{7}$$

Learning cost maps that reflect such cost margins only requires expert demonstrations, which are readily available in archival log data. However, sometimes candidates trajectories that are equally distant from the expert one should bear different costs. We provide an example (right) where the red trajectory should cost more than blue in the presence of an obstacle despite both being equidistant from the expert demonstration.



Guided planning: To further distinguish among candidate trajectories, one could introduce extra penalty terms given additional supervision.

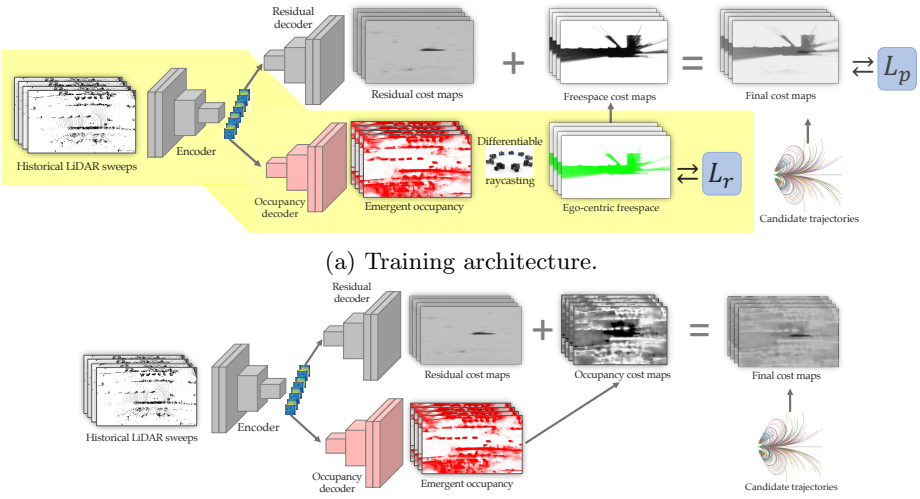
$$D(\hat{\mathbf{y}}_\mathbf{2}, \mathbf{y_2}) = ||\hat{\mathbf{y}}_\mathbf{2}, \mathbf{y_2}||_2 + \gamma\ P(\hat{\mathbf{y}}_\mathbf{2}) \tag{8}$$

where P represents a penalty function and γ is a predefined scaling factor. Zeng et al. [31] propose to define an additional penalty such that candidate trajectories that collide with object boxes would cost an additional γ in addition to the deviation from the expert demonstration. We refer to this approach as *object-guided planning*, which is effective but costly as it requires object track labels.

More scalable alternatives to object supervision can be adopted, such as formulation of the penalty term proposed by Hu et al. [9]. Concretely, candidate trajectories that reach outside the freespace as observed by future LiDAR poses would incur an additional penalty. We refer to this as *freespace-guided planning*.

Residual costmaps: Instead of directly predicting the cost map $\mathbf{c_2}[\mathbf{u}]$, we follow prior work [9] and predict a residual cost map $\tilde{\mathbf{c}}_\mathbf{2}[\mathbf{u}]$ that is added to the cost map from freespace estimate based on predicted emergent occupancy.

$$\mathbf{c_2}[\mathbf{u}] = \tilde{\mathbf{c}}_\mathbf{2}[\mathbf{u}] + \alpha\ \mathrm{proj}(\hat{\mathbf{z}}_\mathbf{2}; \mathbf{y_2})[\mathbf{u}],\ \ \mathbf{u} \in \hat{\mathbf{y}}_\mathbf{2} \tag{9}$$



(a) Training architecture.

(b) Inference architecture.

Fig. 4. Overview of our training and inference-time planning architectures. Highlighted network branch in (a) is used to learn future emergent occupancy, which is augmented by the residual branch that predicts residual cost maps, eventually used in computing a guided planning loss.

where α is a predefined constant and $\tilde{\mathbf{c}}_{\mathbf{2}}$ represents the predicted residual cost map. The operation $\text{proj}(\hat{\mathbf{z}}_{\mathbf{2}}; \mathbf{y}_{\mathbf{2}})$ is illustrated in Fig. 3.

Multi-task planning (new): In addition to the raycasting loss in Fig. 4, we add L_p as an additional planning loss. In other words, the emergent occupancy prediction architecture is augmented with another decoder branch to predict the residual cost maps while sharing the encoder features. Because of this, emergent occupancy forecasting becomes the auxiliary task for the end-to-end motion planner. We illustrate the network architecture during training in Fig. 4 (a).

Test-time occupancy cost maps (new): At test time, to compute ego-centric freespace cost maps based on predicted emergent occupancy, for each candidate sample trajectory, one would need to perform raycasting from its way-points, which is prohibitively expensive. Fortunately, this is exactly equivalent to directly accessing emergent occupancy on the waypoints along the candidate trajectory (because of the cumulative max-operation used in deriving freespace from occupancy - see Fig. 3), as formally expressed in Eq. (10).

$$\text{proj}(\hat{\mathbf{z}}_{\mathbf{2}}; \hat{\mathbf{y}}_{\mathbf{2}})[\mathbf{u}] = \hat{\mathbf{z}}_{\mathbf{2}}[\mathbf{u}], \;\; \mathbf{u} \in \hat{\mathbf{y}}_{\mathbf{2}} \tag{10}$$

The simplified test-time architecture is illustrated in Fig. 4 (b). When optimizing for future trajectories, we restrict the search space of future trajectories to the ones with a smooth transition from the past trajectory [31,9]. Please refer to the supplement for other implementation information such as detailed network architecture.

Dataset	Diff. Raycast	$\frac{\lvert \mathbf{d}-\hat{\mathbf{d}} \rvert}{\mathbf{d}}$ (↓)	BCE (↓)	F1 (↑)	AP (↑)
nuScenes	- [9]	0.297	0.221	0.665	0.769
	✓	**0.242**	**0.140**	**0.777**	**0.863**
ONCE	- [9]	0.371	0.143	0.635	0.732
	✓	**0.243**	**0.097**	**0.787**	**0.827**

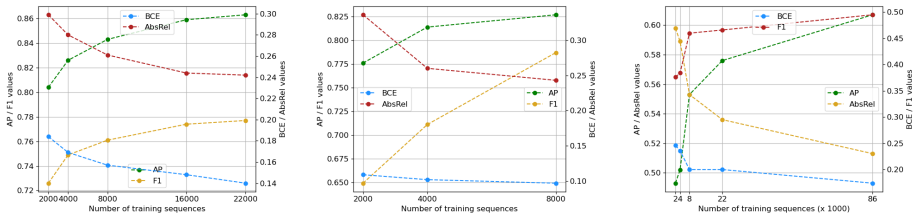
Table 1. Indirect evaluation of emergent occupancy forecasting with respect to groundtruth LiDAR sweeps. On both nuScenes and ONCE, we significantly improve forecasting accuracy across all metrics by using differentiable raycasting for decoupling the scene and ego-motion, unlike Hu *et al.* [9].

4 Experiments

Datasets: We evaluate occupancy forecasting and motion planning on two datasets: nuScenes [2] and ONCE [13]. nuScenes features real-world driving data with 1,000 fully annotated 15 second logs. ONCE is the largest driving dataset with 150 hours of real-world data including 1 million LiDAR sweeps, collected in a range of diverse environments such as urban and suburban areas. As annotation is expensive, only a small subset of logs in ONCE are fully annotated, making it ideal for self-supervised learning. We include comparison against state-of-the-art forecasting and planning approaches on both datasets. We also construct multiple baselines for all ablative evaluation for bird's eye-view motion planning. To understand how our occupancy forecasting and motion planning performance scales to an increasing amount of training data, we randomly curate different training sets of the datasets. Since only a small subset of 8K samples in ONCE is labeled, we do this by progressively increasing the number of training samples by adding scenes from both their labeled and unlabeled-small splits, which include 8K, and 86K training samples respectively. Some of our analysis exists only on the combined labeled and unlabeled-small split which totals to 94K samples. For nuScenes, we randomly sample scenes from their official training set. For all experiments that follow, we take in a historical LiDAR stack of 2 seconds and forecast for the next 3 seconds.

4.1 Emergent Occupancy Forecasting

Metrics: Since, the groundtruth for true occupancy is unavailable, we quantitatively evaluate the LiDAR sweeps raycast from the emergent occupancy predictions. Specifically, our first evaluation computes the absolute relative error between the groundtruth distance traveled by every ray starting from the sensor origin, and the expected distance traveled by corresponding rays; where the expected distance is obtained by casting rays through the forecasted occupancy. Second, we score *every* BEV voxel traversed by a ray using its 'free' or 'not-free' state. This dense per-ray evaluation is equivalent to evaluating the per-pixel binary classification of an ego-centric freespace map with respect to



(a) nuScenes (b) ONCE-labeled (c) ONCE-unlabeled

Fig. 5. We highlight the merits of our self-supervised approach which can be given any amount of unlabeled LiDAR data to train on, in the form of posed archival LiDAR sweeps, thereby increasing the performance of emergent occupancy forecasting (evaluated using classification metrics such as average-precision and F1). Please refer to the supplement for corresponding tables.

its groundtruth, allowing us to compare to the baseline discussed below. We compute the dense binary cross-entropy, average precision and the F1-score. All metrics are averaged across all prediction timesteps (up to 3s).

Baseline: We re-implement the future-freespace architecture from [9] which directly forecasts ego-centric freespace. For building our architecture, we adapt this network to predict an arbitrary quantity which differentiably raycasts into ego-centric freespace given a sensor location. On training this architecture in a self-supervised manner, the arbitrary quantity *emerges* into emergent occupancy, an explainable intermediate representation for downstream motion planners.

Main results: We compare the performance of both approaches in Tab. 1. Note the drastic improvement in all metrics on using differentiable raycasting to decouple the scene motion from the ego-motion of the sensor on both nuScenes and ONCE. With increase of up to 15% F1 points, we highlight the high-quality of our predicted occupancy and the pronounced effect of adding differentiable raycasting. Our results show that occupancy reasoning is an important intermediate task, *even* if the end-goal is simply understanding freespace: Our method, which predicts occupancy as an intermediate target, outperforms [9], which directly aims to predict freespace. Fig. 6 visualizes predicted ego-centric freespace for a single scenario in ONCE using [9] and our approach at $t = 0, 3s$ in the future. In Fig. 5, we show how adding more training samples to both datasets result in an upward trend in performance across *all* metrics. This increasing generalizability and scaling of training data comes for free with our self-supervised approach.

4.2 Motion Planning

Metrics: We follow prior works and compute three metrics for evaluating motion planning performance, including (1) L2 error; (2) point collision rate; (3) box collision rate. The L2 distance measures how close the planned trajectory follows the expert trajectory at each future timestamp. The point collision rate measures how often the planned waypoint is within the BEV boxes of other objects. The

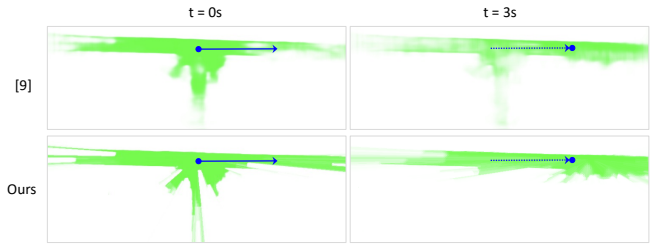


Fig. 6. Future ego-centric freespace from [9] and our model, raycasted from predicted emergent occupancy. Note how the presence of moving and parked cars on roadsides is captured well by our approach even 3s in the future.

nuScenes	Box Collision (%)			L2 Error (m)		
	1s	2s	3s	1s	2s	3s
IL [21]	0.08	0.27	1.95	**0.44**	**1.15**	**2.47**
FF [9]	0.06	0.17	1.07	0.55	1.20	2.54
Ours	**0.04**	**0.09**	**0.88**	0.67	1.36	2.78
NMP [31]	0.04	0.12	*0.87*	*0.53*	*1.25*	*2.67*
P3 [23]	*0.00*	*0.05*	1.03	0.59	1.34	2.82

Table 2. We compare end-to-end state-of-the-art motion planners on nuScenes-val. NMP and P3 are supervised approaches that have access to object tracking labels.

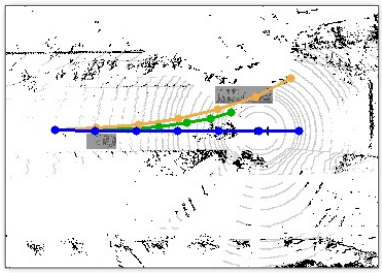
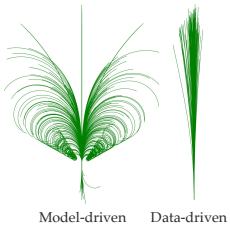


Fig. 7. A vanilla spacetime trajectory with a lower L2 error wrt. expert, may collide into objects unlike a proposed trajectory with larger L2 error but no collision.

box collision rate measures how often the BEV box of the ego-vehicle intersects with BEV boxes of other objects.

Trajectory sampling: When evaluating performance on nuScenes, we follow previous state-of-the-art approaches [31,9] and sample a combination of straight lines, circles, and clothoid curves as trajectory samples. Owing to the scene diversity in ONCE, we notice that such a sampling strategy does not capture the distribution of expert trajectories on ONCE as they range widely in their velocities and directions. Inspired by [3], we sample a data-driven trajectories to complement the model-driven samples (right). The supplement provides more details on our data-driven sampler.



Planning on nuScenes

Baselines: We compare our proposed approach to four baseline end-to-end motion planners. First, we implement a pure imitation learning (IL) baseline, a max-margin neural motion planner self-supervised by expert trajectories, as described

	Freespace Guided	Multi Task	Diff. Raycast	Box Collision (%)			Point Collision (%)			L2 Error (m)		
				1s	2s	3s	1s	2s	3s	1s	2s	3s
(a)	-	-	-	0.08	0.27	1.95	**0.00**	**0.00**	0.35	0.44	1.15	2.47
(b)	✓	-	-	0.06	0.17	1.07	**0.00**	0.01	0.04	0.55	1.20	2.54
(c)	-	✓	-	0.08	0.17	1.29	**0.00**	0.02	0.08	**0.42**	**1.06**	**2.30**
(d)	✓	✓	-	**0.02**	0.10	1.10	**0.00**	**0.00**	0.08	0.52	1.22	2.64
(e)	✓	✓	✓	0.04	**0.09**	**0.88**	**0.00**	0.01	**0.03**	0.67	1.36	2.78

Table 3. Ablation studies on nuScenes-val. Note that (a) is IL, (b) is FF, and (e) is **Ours** in Tab. 2.

in Eq. (7). Second, we re-implement future-freespace-guided max-margin planner (FF) proposed by Hu *et al.* [9], as captured by Eq. (8). Third, we re-implement a simplified neural motion planner (NMP) without modeling costs related to map information and traffic light status as such information is unavailable on nuScenes. Last, we re-implement a simplified version of perceive, predict, and plan (P3) where we do not distinguish semantic occupancy of different classes. To ensure a fair comparison, we adopt the same neural net architecture for the baselines and our approach.

Main results: As Tab. 2 shows, in terms of collision rates, our self-supervised approach outperforms both self-supervised baselines (IL and FF) by a large margin. Moreover, our approach achieves the same collision rate at 3s as the best of supervised baselines. We also observe a commonly observed trade-off between L2 errors and collision rates [31]. For example, pure imitation learning achieves the lowest L2 errors with the highest collision rates.

Ablation studies: We perform extensive ablation studies in Tab. 3 to understand where improvements come from. There are three main observations:

- Differentiable raycasting reduces collision rate at further horizon (3s), as seen in (d) vs. (e), suggesting decoupling motion of the world (space-time occupancy) from ego-motion is helpful when learning long range cost maps.
- Multi-task learning further reduces collision rates, as seen in (a) vs. (c). Training max-margin planners with an auxiliary self-supervised forecasting task significantly reduces the collision rates without hurting L2.
- Freespace-guided cost margin is crucial to lowering collision rates, as seen in (a) vs. (b), (c) vs. (d). However, there is a trade-off: the L2 errors tend to increase as being expert-like (at all costs) is no longer the only objective. In Fig. 7, we show an example result describing why L2 error is a misleading metric that doesn't allow for alternate future plans that are otherwise viable. Additionally, Casas *et al.* [3] show that collision rate is a more consistent metric between evaluation in the open- and closed-loop setups.

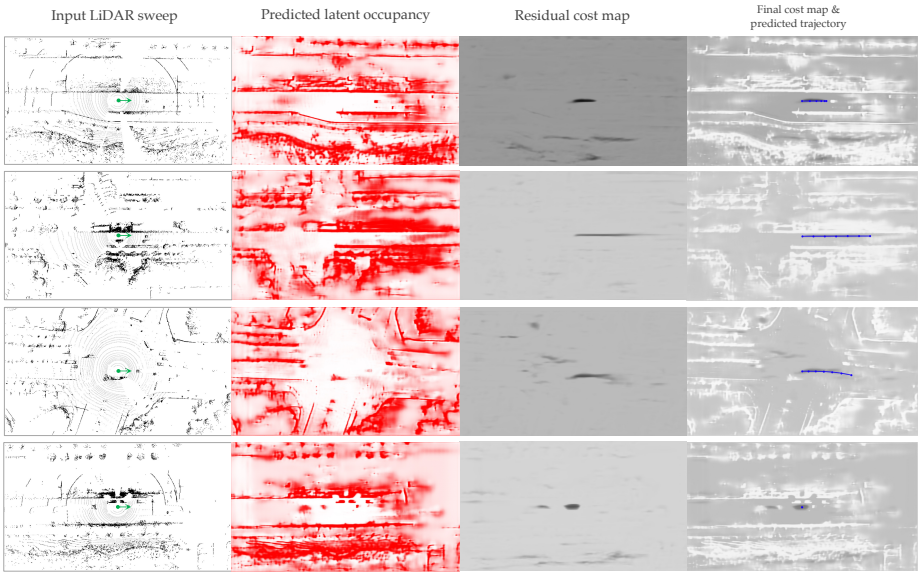


Fig. 8. Qualitative results of our learned model. From top to bottom, we visualize various scenarios, including slowing down, speeding up, navigating an intersection and staying still. All columns after the first one are visualized at future timestamp t=0.5s. We successfully forecast the motion of surrounding objects, e.g. in third row, which results in safer planned trajectories.

Planning on ONCE

Baseline: ONCE offers a massive amount of unlabeled, diverse LiDAR sweeps paired with ego-vehicle trajectories and a small fully labeled subset of about 8K samples. We train a re-implemented neural motion planner as a supervised baseline on the fully labeled subset. We train our self-supervised approach over a wide range of training sizes, from 2K to 94K.

Main results: Perhaps unsurprisingly, our first observation is that the metrics on ONCE are inflated as compared to nuScenes, because of the diverse range of environments ONCE features, ranging from straight highways to complex city road structures. To show the scalability of our approach on such a diverse and large dataset, we plot the L2 error and (box) collision rate at 3s as a function of the amount of training data in Fig. 10. Both the L2 error and the collision rate of our approach continue to improve as we increase the size of the training set. In com-

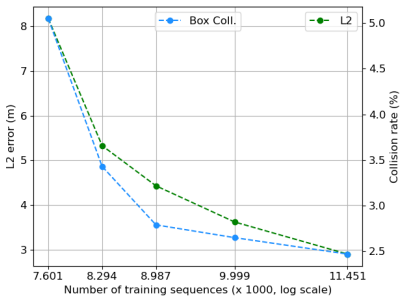


Fig. 10. Planning performance vs. larger ONCE training set size.

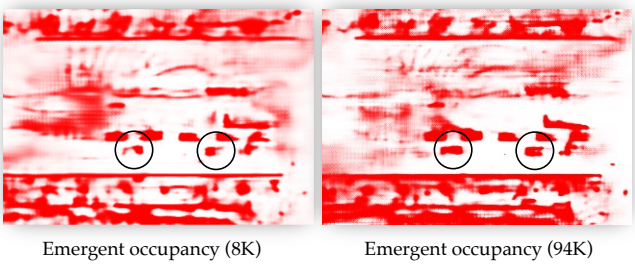


Fig. 9. Evolution of estimated emergent future occupancy.

parison, the supervised neural motion planner achieves an L2 error of 4.45m and a box collision rate of 2.54% at a training size of 8K.

At 94K training samples, our self-supervised approach achieves a dramatically lower L2 error of 2.9m and a lower collision rate of 2.47%. Importantly, such scalability for motion planning comes for free as our approach is self-supervised. We show some qualitative results on the ONCE dataset in Fig. 8 where our approach is able to deal with a number of varying driving scenarios; decelerate and stop when necessary, predict long trajectories when unoccupied regions are predicted ahead, avoid collisions with other vehicles while navigating an intersection, or stay stationary. Please refer to our supplement for further quantitative evaluation, visualization of future cost maps and more qualitative examples that feature failure cases (e.g., forecasted occupancy diffuses over time).

Evolution of occupancy estimates: Our model tends to produce better estimates of emergent occupancy as we increase the amount of training data. The percent of semantic object pixels recalled from the ground-truth semantic object labels in our predicted occupancy map increases from 51% to 59% at t=0s when we increase the amount of training data from 8K to 94K. Qualitatively, this can be seen in Fig. 9 where the shape of two cars in the right lane looks more "space-time complete" for the model trained with increased data.

5 Conclusion

We propose *emergent occupancy* as a self-supervised and explainable representation for motion planning. Our novel differentiable raycasting procedure enables the learning of occupancy forecasting under the self-supervised task of LiDAR sweep forecasting. The raycasting setup also allows us to decouple ego motion from scene motion, making forecasting an easier task for the network to learn. Experimental results suggest that such decoupling is also helpful for downstream motion planning. Such training at scale allows object shape, tracks, and multiple futures to "emerge" in the predicted emergent occupancy.

Acknowledgments This work was supported by the CMU Argo AI Center for Autonomous Vehicle Research.

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 6
2. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving. arXiv preprint arXiv:1903.11027 (2019) 9
3. 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) 4, 11, 12
4. Chai, Y., Sapp, B., Bansal, M., Anguelov, D.: Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. arXiv preprint arXiv:1910.05449 (2019) 3
5. Chen, D., Zhou, B., Koltun, V., Krähenbühl, P.: Learning by cheating. In: Conference on Robot Learning. pp. 66–75. PMLR (2020) 4
6. Codevilla, F., Miiller, M., López, A., Koltun, V., Dosovitskiy, A.: End-to-end driving via conditional imitation learning. In: 2018 IEEE International Conference on Robotics and Automation. pp. 1–9. IEEE (2018) 4
7. Codevilla, F., Santana, E., López, A.M., Gaidon, A.: Exploring the limitations of behavior cloning for autonomous driving. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 9329–9338 (2019) 4
8. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: Carla: An open urban driving simulator. arXiv preprint arXiv:1711.03938 (2017) 4
9. 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, 2, 3, 6, 7, 8, 9, 10, 11, 12
10. Hu, P., Ziglar, J., Held, D., Ramanan, D.: What you see is what you get: Exploiting visibility for 3d object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11001–11009 (2020) 3
11. Lang, A.H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O.: Pointpillars: Fast encoders for object detection from point clouds. In: CVPR (2019) 3
12. Lombardi, S., Simon, T., Saragih, J., Schwartz, G., Lehrmann, A., Sheikh, Y.: Neural volumes: Learning dynamic renderable volumes from images. arXiv preprint arXiv:1906.07751 (2019) 4
13. Mao, J., Niu, M., Jiang, C., Liang, H., Liang, X., Li, Y., Ye, C., Zhang, W., Li, Z., Yu, J., et al.: One million scenes for autonomous driving: Once dataset. arXiv preprint arXiv:2106.11037 (2021) 3, 9
14. 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) 4
15. Mittal, H., Okorn, B., Held, D.: Just go with the flow: Self-supervised scene flow estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2020) 1, 3, 4
16. Montemerlo, M., Becker, J., Bhat, S., Dahlkamp, H., Dolgov, D., Ettinger, S., Haehnel, D., Hilden, T., Hoffmann, G., Huhnke, B., et al.: Junior: The stanford entry in the urban challenge. Journal of field Robotics **25**(9), 569–597 (2008) 3
17. Osa, T., Pajarinen, J., Neumann, G., Bagnell, J.A., Abbeel, P., Peters, J.: An algorithmic perspective on imitation learning. arXiv preprint arXiv:1811.06711 (2018) 4

18. Park, K., Sinha, U., Barron, J.T., Bouaziz, S., Goldman, D.B., Seitz, S.M., Martin-Brualla, R.: Nerfies: Deformable neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5865–5874 (2021) 4
19. Pomerleau, D.A.: Alvinn: An autonomous land vehicle in a neural network. In: Advances in neural information processing systems. pp. 305–313 (1989) 4
20. Qi, C.R., Zhou, Y., Najibi, M., Sun, P., Vo, K., Deng, B., Anguelov, D.: Offboard 3d object detection from point cloud sequences. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6134–6144 (2021) 4
21. Ratliff, N.D., Bagnell, J.A., Zinkevich, M.A.: Maximum margin planning. In: Proceedings of the 23rd international conference on Machine learning. pp. 729–736 (2006) 11
22. Rhinehart, N., McAllister, R., Levine, S.: Deep imitative models for flexible inference, planning, and control. arXiv preprint arXiv:1810.06544 (2018) 4
23. Sadat, A., Casas, S., Ren, M., Wu, X., Dhawan, P., Urtasun, R.: Perceive, predict, and plan: Safe motion planning through interpretable semantic representations. European Conference on Computer Vision (2020) 2, 3, 4, 11
24. Sadat, A., Ren, M., Pokrovsky, A., Lin, Y.C., Yumer, E., Urtasun, R.: Jointly learnable behavior and trajectory planning for self-driving vehicles. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 3949–3956. IEEE (2019) 3
25. Urmson, C., Anhalt, J., Bagnell, D., Baker, C., et al.: Autonomous driving in urban environments: Boss and the urban challenge. Journal of Field Robotics **25**(8), 425–466 (2008) 3, 4
26. Weng, X., Kitani, K.: A baseline for 3d multi-object tracking. arXiv preprint arXiv:1907.03961 **1**(2), 6 (2019) 3
27. Weng, X., Wang, J., Levine, S., Kitani, K., Rhinehart, N.: 4d forecasting: Sequential forecasting of 100,000 points (2020) 4
28. Weng, X., Wang, J., Levine, S., Kitani, K., Rhinehart, N.: Inverting the pose forecasting pipeline with spf2: Sequential pointcloud forecasting for sequential pose forecasting. arXiv preprint arXiv:2003.08376 (2020) 1, 3, 4
29. Wilson, B., Qi, W., Agarwal, T., Lambert, J., Singh, J., Khandelwal, S., Pan, B., Kumar, R., Hartnett, A., Pontes, J.K., 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) 1, 3, 4
30. Yen-Chen, L., Florence, P., Barron, J.T., Rodriguez, A., Isola, P., Lin, T.Y.: inerf: Inverting neural radiance fields for pose estimation. arXiv preprint arXiv:2012.05877 (2020) 4
31. 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) 4, 7, 8, 11, 12