Point Cloud Forecasting as a Proxy for 4D Occupancy Forecasting

Tarasha Khurana* Peiyun Hu* David Held Deva Ramanan

Carnegie Mellon University

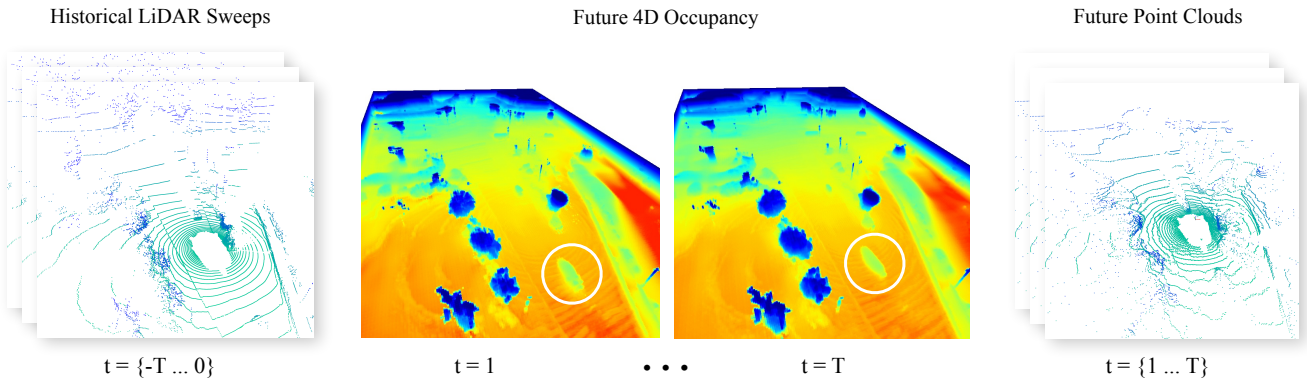

Figure 1. We focus on the problem of scene perception and forecasting for autonomous systems. As traditional methods rely on costly human annotations, we look towards emerging self-supervisable and scalable tasks such as point cloud forecasting [11, 18, 19]. However, we argue that the formulation of point cloud forecasting unnecessarily focuses on learning the sensor extrinsics and intrinsics as part of predicting future point clouds, whereas the only physical quantity of central importance to autonomous perception is future *spacetime 4D occupancy*. We recast the task as that of 4D occupancy forecasting and show how using the same data as point cloud forecasting, one can learn a meaningful and generic intermediate quantity – future spacetime 4D occupancy.

Abstract

Predicting how the world can evolve in the future is crucial for motion planning in autonomous systems. Classical methods are limited because they rely on costly human annotations in the form of semantic class labels, bounding boxes, and tracks or HD maps of cities to plan their motion — and thus are difficult to scale to large unlabeled datasets. One promising self-supervised task is 3D point cloud forecasting [11, 18–20] from unannotated LiDAR sequences. We show that this task requires algorithms to implicitly capture (1) sensor extrinsics (i.e., the egomotion of the autonomous vehicle), (2) sensor intrinsics (i.e., the sampling pattern specific to the particular LiDAR sensor), and (3) the shape and motion of other objects in the scene. But autonomous systems should make predictions about the world and not their sensors! To this end, we factor out (1) and (2) by recasting the task as one of spacetime (4D) occupancy forecasting. But because it is expensive to obtain ground-truth 4D occupancy, we "render" point cloud data from 4D occupancy predictions given sensor extrinsics and intrinsics, allowing one to train and test occupancy algorithms with unannotated LiDAR sequences. This also allows one to evaluate and compare point cloud forecasting algorithms across diverse datasets, sensors, and vehicles.

1. Introduction

Motion planning in a dynamic environment requires autonomous agents to predict the motion of other objects. Standard solutions consist of perceptual modules such as mapping, object detection, tracking, and trajectory forecasting. Such solutions often rely on human annotations in the form of HD maps of cities, or semantic class labels, bounding boxes, and object tracks, and therefore are difficult to scale to large unlabeled datasets. One promising *self-supervised* task is 3D point cloud forecasting [11, 18–20].

*Equal contribution

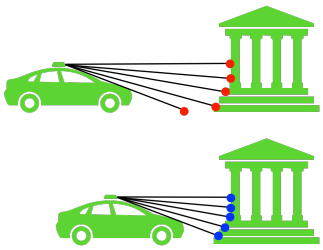

Figure 2. Points depend on the intersection of rays from the depth sensor and the environment. Therefore, accurately predicting points requires accurately predicting sensor extrinsics (sensor egomotion) and intrinsics (ray sampling pattern). But we want to understand dynamics of the environment, not our LiDAR sensor!

Since points appear where lasers from the sensor and scene intersect, the task of forecasting point clouds requires algorithms to implicitly capture (1) sensor extrinsics (*i.e.*, the ego-motion of the autonomous vehicle), (2) sensor intrinsics (*i.e.*, the sampling pattern specific to the LiDAR sensor), and (3) the shape and motion of other objects in the scene. This task can be non-trivial even in a static scene (Fig. 2). We argue that autonomous systems should focus on making predictions about the world and not themselves, since an ego-vehicle has access to its future motion plans (extrinsics) and calibrated sensor parameters (intrinsics).

We factor out these (1) sensor extrinsics and (2) intrinsics by recasting the task of point cloud forecasting as one of spacetime (4D) occupancy forecasting. This disentangles and simplifies the formulation of point cloud forecasting, which now focuses solely on forecasting the central quantity of interest, the 4D occupancy. Because it is expensive to obtain ground-truth 4D occupancy, we "render" point cloud data from 4D occupancy predictions given sensor extrinsics and intrinsics. In some ways, our approach can be seen as the spacetime analog of novel-view synthesis from volumetric models such as NeRFs [12]; rather than rendering images by querying a volumetric model with rays from a known camera view, we render a LiDAR scan by querying a 4D model with rays from known sensor intrinsics and extrinsics. This allows one to train and test 4D occupancy forecasting algorithms with un-annotated LiDAR sequences. This also allows one to evaluate and compare point cloud forecasting algorithms across diverse datasets, sensors, and vehicles. We find that our approach to 4D occupancy forecasting, which can also render point clouds, performs drastically better than SOTAs in point cloud forecasting, both quantitatively (by up to 3.26m L1 error, Tab. 1) and qualitatively (Fig. 6). Our method beats prior art with zero-shot cross-sensor generalization (Tab. 2). To our knowledge, these are first results that generalize across train/test sensor rigs, illustrating the power of disentangling sensor motion from scene motion.

2. Related Work

Point Cloud Forecasting As one of the most promising self-supervised tasks that exploit unannotated LiDAR sequences, point cloud forecasting [11, 18–20] provides the algorithm past point clouds as input and asks it to predict future point clouds as output. Traditionally, both the input and the output are defined in the sensor coordinate frame, which moves with time. Although this simplifies preprocessing by eliminating the need for a local alignment, it forces the algorithm to implicitly capture (1) sensor extrinsics (i.e., the egomotion of the autonomous vehicle), (2) sensor intrinsics (i.e., the sampling pattern specific to the particular LiDAR sensor), and (3) the shape and motion of other objects in the scene. We argue that autonomous systems should make predictions about the world and not their sensors. In this paper, we reformulate point cloud forecasting by factoring out sensor extrinsics and intrinsics. Concretely, the new setup asks the algorithm to estimate the depth for rays from future timestamps. We show that one could use it as a proxy for training and testing 4D occupancy forecasting algorithms. Moreover, we demonstrate that one can evaluate existing point cloud forecasting methods under this setup, allowing 4D occupancy forecasting algorithms to be compared with point cloud forecasting algorithms.

Occupancy Forecasting Occupancy, as a predictive representation complementary to standard object-centric representations in the context of supporting downstream motion planning, has gained popularity over the last few years due to its efficiency in representing complex scenarios and interactions. Most existing works on occupancy forecasting focus on *semantic* occupancy grids from a bird's-eye view (BEV) [5, 10, 16]. They choose to focus on 2D for a good reason since most autonomous driving planners reason in a 2D BEV space. A downside is that it is expensive to obtain ground-truth *semantic* BEV occupancy for training and testing algorithms. [7] claim that if we reduce our goal from *semantic* occupancy to *geometric* occupancy, that is knowing if a location is occupied without asking which type of object is occupying it, one could learn to forecast *geometric* BEV occupancy from unannotated LiDAR sequences. In this paper, we take the idea from [7] and go beyond BEV – we propose an approach to learning to forecast 4D *geometric* occupancy from unannotated LiDAR sequences. We also propose a scalable evaluation to this task that admits standard point cloud forecasting methods.

Novel View Synthesis We have seen tremendous progress in novel view synthesis in the last few years [9, 12, 13]. At its core, the differentiable nature of volumetric rendering allows one to optimize the underlying 3D structure of the scene by fitting samples of observations with known sen-

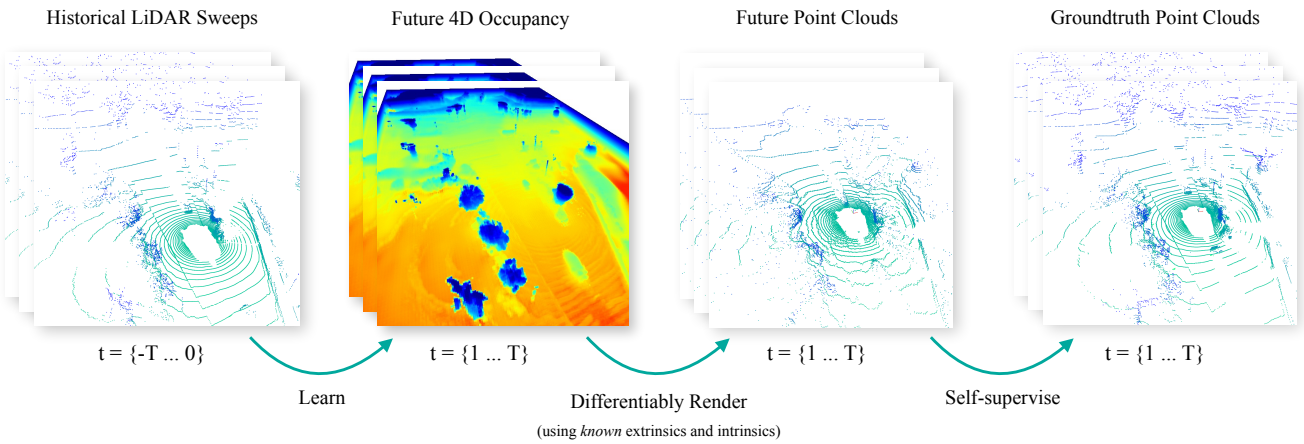

Figure 3. High-level overview of the approach we follow, closely inspired by a prior work [7]. Instead of directly predicting future point clouds by observing a set of historical point clouds, we take a geometric perspective on this problem and instead forecast a generic intermediate 3D occupancy-like quantity within a bounded volume. Known sensor extrinsics and intrinsics are an input to our method, which is different from how classical point cloud forecasting is formulated. We argue that this factorization is sensible as an autonomous agent plans its own motion and has access to sensor information. Please refer to our supplement for architectural details.

sor poses without explicit 3D supervision. Our work can be thought of as novel view synthesis, where we try to synthesize depth images from novel views at future timestamps. Thanks to motion sensors (e.g., IMU), one can assume that relative LiDAR pose among frames in a log can be reliably estimated. Our work also differs from common novel view synthesis literatures in a few important aspects: (a) we use an efficient feed-forward network to predict the spacetime occupancy volume instead of applying test-time optimization; (b) we optimize an explicit volumetric scene representation (i.e., occupancy grid) instead of an implicit neural scene representation; (c) our approach relies on shape and motion prior learned across diverse scenarios in order to predict what happens next instead of reconstructing based on samples only from a specific scenario.

3. Method

Autonomous fleets log an abundance of unannotated sequences of LiDAR point clouds $\mathbf{X}_{-T...T}$, where we also estimate the relative sensor location for each frame $\mathbf{o}_{-T:T}$. Suppose we split such a sequence into a historic part $\mathbf{X}_{-T:0}$ and $\mathbf{o}_{-T:0}$ and a future part $\mathbf{X}_{1:T}$ and $\mathbf{o}_{1:T}$.

Standard point cloud forecasting methods, denoted by function g, take the historical sequence of point clouds $\mathbf{X}_{-T:0}$ as input and try to predict the future sequence of point clouds $\hat{\mathbf{X}}_{1:T}$.

$$\hat{\mathbf{X}}_{1:T} = g(\mathbf{X}_{-T:0}) \tag{1}$$

To introduce our approach, we need to first reparametrize a point from the future LiDAR point cloud, say $\mathbf{x} \in \mathbf{X}_t$ where $t = 1 \dots T$, as a ray that starts from the sensor location $\mathbf{o}_t$, travels along the direction $\mathbf{d}$, and reaches the end point $\mathbf{x}$ after a distance of λ:

$$\mathbf{x} = \mathbf{o}_t + \lambda \mathbf{d}, \mathbf{x} \in \mathbf{X}_t \tag{2}$$

Conceptually, our approach, denoted by function f, takes a ray from a future timestamp t parametrized by its origin and direction $(\mathbf{o}_t, \mathbf{d})$, and tries to predict the distance $\hat{\lambda}$ the ray would travel, based on historic sequence of point clouds $\mathbf{X}_{-T:0}$ and sensor locations $\mathbf{o}_{-T:0}$.

$$\hat{\lambda} = f(\mathbf{o}_t, \mathbf{d}; \mathbf{X}_{-T:0}, \mathbf{o}_{-T:0}) \tag{3}$$

Intuitively, Eq. (3) is similar to view synthesis in NERF [12] except we are computing expected depth rather than expected color. Below, we introduce how we formulate the differentiable volumetric rendering process and use it for learning to forecast 4D occupancy.

Spacetime (4D) occupancy We define spacetime occupancy as the occupied state of a 3D location at a particular time instance. We use $\mathbf{z}$ to denote the true spacetime occupancy, which may not be directly observable due to line-of-sight visibility constraints. Consider a bounded spatial-temporal 4D volume, $\mathcal{V}$, which is discretized into spacetime voxels $\mathbf{v}$. We can use

$$\mathbf{z}[\mathbf{v}] \in \{0, 1\}, \mathbf{v} = (x, y, z, t), \mathbf{v} \in \mathcal{V} \tag{4}$$

to represent the occupancy of voxel $\mathbf{v}$ in the spacetime voxel grid $\mathcal{V}$, which can be *occupied* (1) or *free* (0).

In practice, we learn an occupancy predictio network h (parametrized by $\mathbf{w}$) to predict discretized spacetime 4D occupancy given historic sequence of point clouds and sensor locations,

$$\hat{\mathbf{z}} = h(\mathbf{X}_{-T:0}, \mathbf{o}_{-T:0}; \mathbf{w}) \tag{5}$$

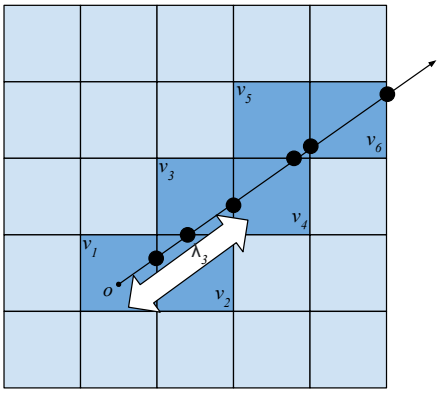

Figure 4. We illustrate the process of rendering depth for a given ray from the predicted occupancy grid. We assume that rays only stop at the voxel boundary, which discretizes the output space into a discrete set of events. We then compute the probability for a ray stopping at each boundary intersection. Finally, we compute the expected stopping distance.

where

$$\hat{\mathbf{z}}[\mathbf{v}] \in \mathbb{R}_{[0,1]} \tag{6}$$

represents the predicted occupancy of voxel $\mathbf{v}$ in the space-time voxel grid $\mathcal{V}$. Please refer to the supplementary materials for network architecture details.

Depth rendering from occupancy Given a ray query $\mathbf{x} = \mathbf{o} + \lambda\mathbf{d}$, our goal is to predict $\hat{\lambda}$ as close to λ as possible. We first compute how it intersects with the occupancy grid by voxel traversal [2] (Fig. 4). Suppose the ray intersects with a list of voxels $\{\mathbf{v}_1 \dots \mathbf{v}_n\}$. We discretize the ray space by assuming that a ray can only stop at voxel boundaries or infinity. We interpret occupancy of voxel $\mathbf{v}_i$ as the conditional probability that a ray leaving voxel $\mathbf{v}_{i-1}$ would stop in voxel $\mathbf{v}_i$. We can write

$$p_i = \prod_{j=1}^{i-1}(1 - \hat{\mathbf{z}}[\mathbf{v}_j])\hat{\mathbf{z}}[\mathbf{v}_i] \tag{7}$$

where p_i represents the probability that a ray stops in voxel $\mathbf{v}_i$. Now we can render the distance by computing the stopping point in expectation.

$$\hat{\lambda} = f(\mathbf{o}, \mathbf{d}) = \sum_{i=1}^{n} p_i \hat{\lambda}_i \tag{8}$$

where $\hat{\lambda}_i$ represents the stopping distance at voxel $\mathbf{v}_i$.

You may have noticed that Eq. (8) does not capture the case where the ray stops outside the voxel grid, where the stopping distance is ill-defined (it will stop at infinity). During training, we allow a virtual stopping point outside the grid at the ground-truth location, i.e.,

$$\hat{\lambda} = f(\mathbf{o}, \mathbf{d}) = \sum_{i=1}^{n} p_i \hat{\lambda}_i + \prod_{i=1}^{n}(1 - p_i)\hat{\lambda}_{n+1} \tag{9}$$

where $\hat{\lambda}_{n+1} = \lambda$.

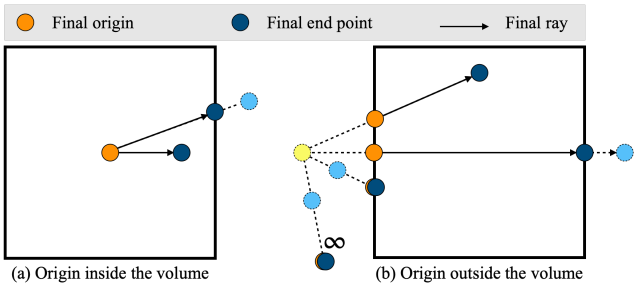

Figure 5. Ray Clamping. First, we move the origin towards the end point until the origin touches the volume or infinity. Then, we move the end point towards the origin until the end point touches the volume or infinity. At all times, we make sure the end point stays ahead of the origin (like two rings on a string). Being inside the volume counts as touching it.

Loss function We can train the occupancy prediction network with a simple L1 loss between the rendered depth $\hat{\lambda}$ and the ground-truth depth λ.

$$L(\mathbf{w}) = \sum_{(\mathbf{o},\lambda,\mathbf{d})\in(X_{1:T},\mathbf{o}_{1:T})} |\lambda - f(\mathbf{o}, \mathbf{d}; \mathbf{X}_{-T:0}, \mathbf{o}_{-T:0}, \mathbf{w})| \tag{10}$$

4. Evaluation

The golden standard for evaluating 4D occupancy forecasting would be to compare the predicted occupancy with the ground-truth, but because it is extremely expensive to obtain ground-truth 4D occupancy, we "render" future point clouds from forecasted 4D occupancy with known sensor intrinsics and extrinsics, use the quality of rendered future point clouds as a proxy for that of forecasted 4D occupancy.

We introduce a new evaluation, where we factor out sensor intrinsics and extrinsics such that algorithms can be evaluated solely based on how well it captures how the scene unfolds. We provide future rays as queries and ask algorithms to provide a depth estimate for each query.

Given a query ray $\overrightarrow{OQ}$, there is a prediction ray $\overrightarrow{OP}$, where O represents the origin, Q represents the ground-truth end point, and P represents the predicted end point.

$$\overrightarrow{OQ} = \mathbf{o} + \lambda\mathbf{d} \tag{11}$$

$$\overrightarrow{OP} = \mathbf{o} + \hat{\lambda}\mathbf{d} \tag{12}$$

Given such a pair of rays, we define the error ε:

$$\varepsilon = |\overrightarrow{OQ} - \overrightarrow{OP}| = |\overrightarrow{PQ}| = |\lambda - \hat{\lambda}| \tag{13}$$

Near-field error Since LiDAR rays only travel through freespace and terminate when reaching occupied surface, there is a physical meaning behind the ε in Eq. (13). In practice, occupancy and freespace prediction is only relevant in regions that are reachable by the autonomous vehicle in planning's time hoziron. To reflect the focus on the reachable regions, we propose an operation to clamp any given ray $\overrightarrow{XY}$ to the fixed volume $\mathcal{V}$. We call it *ray clamping*, denoted as $\phi_{\mathcal{V}} : \overrightarrow{XY} \rightarrow \overrightarrow{X'Y'}$ and illustrated in Fig. 5.

We define the near-field (bounded by volume $\mathcal{V}$) prediction error $\varepsilon_{\mathcal{V}}$ as

$$\varepsilon_{\mathcal{V}} = |\phi_{\mathcal{V}}(\overrightarrow{OQ}) - \phi_{\mathcal{V}}(\overrightarrow{OP})| = |\overrightarrow{O'Q'} - \overrightarrow{O'P'}| = |\overrightarrow{P'Q'}| \tag{14}$$

Even though this metric penalizes disagreements of predicted depth along query rays within the bounded volume, it does not capture the severity of a prediction error. In real-world, one meter of an error close to the AV matters more. To this end, we also propose using a relative near-field prediction error $\varepsilon_{\mathcal{V}}^{rel}$ defined as,

$$\varepsilon_{\mathcal{V}}^{rel} = \frac{|\phi_{\mathcal{V}}(\overrightarrow{OQ}) - \phi_{\mathcal{V}}(\overrightarrow{OP})|}{|\overrightarrow{OQ}|} = \frac{|\overrightarrow{P'Q'}|}{|\overrightarrow{OQ}|} \tag{15}$$

The proposed evaluation requires one predicted ray for every ground-truth ray (query). Any algorithms that are capable of rendering depth for a given ray by design meets this requirement, including 4D occupancy forecasting from Sec. 3. However, for point cloud forecasting algorithms, the number of predicted points does not necessarily match the number of ground-truth rays, plus there is no one-to-one mapping between predicted and ground-truth points. To resolve this discrepancy, we propose to fit a surface to the predicted point clouds, on which we can query each ground-truth ray, find its intersection with the fitted surface, and output the (clamped) ray distance. In practice, we interpolate depth among the spherical projections of predicted rays.

We also consider vanilla chamfer distance d (16) and near-field chamfer distance $d_{\mathcal{V}}$ (17)

$$d = \frac{1}{2N}\sum_{\mathbf{x}\in\mathbf{X}} \min_{\hat{\mathbf{x}}\in\hat{\mathbf{X}}} ||\mathbf{x}-\hat{\mathbf{x}}||_2^2 + \frac{1}{2M}\sum_{\hat{\mathbf{x}}\in\hat{\mathbf{X}}} \min_{\mathbf{x}\in\mathbf{X}} ||\mathbf{x}-\hat{\mathbf{x}}||_2^2 \tag{16}$$

where $\mathbf{X}$, $\hat{\mathbf{X}}$ represents the ground-truth, predicted point cloud; N and M are their respective number of points.

$$d_{\mathcal{V}} = \frac{1}{2N'}\sum_{\mathbf{x}\in\hat{\mathbf{X}_{\mathcal{V}}}} \min_{\hat{\mathbf{x}}\in\mathbf{X}_{\mathcal{V}}} ||\mathbf{x}-\hat{\mathbf{x}}||_2^2 + \frac{1}{2M'}\sum_{\mathbf{x}\in\mathbf{X}_{\mathcal{V}}} \min_{\hat{\mathbf{x}}\in\hat{\mathbf{X}}_{\mathcal{V}}} ||\mathbf{x}-\hat{\mathbf{x}}||_2^2 \tag{17}$$

where $\mathbf{X}_{\mathcal{V}}$, $\hat{\mathbf{X}}_{\mathcal{V}}$ represents the ground-truth point cloud and predicted point cloud within the bounding volume $\mathcal{V}$; M', N' are their respective number of points.

5. Experiments

Datasets We perform experiments on nuScenes [4], KITTI-Odometry [3, 6] and ArgoVerse2.0 [20]. nuScenes [4] is a full-suite autonomous driving dataset with a total of 1,000 real-world driving sequences of 15s each. KITTI [6] is also a multi-sensor dataset with 6 hours of diverse driving data across freeways and urban areas. KITTI-Odometry is a subset of this KITTI dataset where sequences have accurate sensor poses. ArgoVerse2.0 [20] contains the largest set of unannotated LiDAR sequences. Please see the supplementary material for results on ArgoVerse2.0.

Setup We consider a bounded area around the autonomous vehicle: -70m to 70m in the x-axis, -70m to 70m in the y-axis and -4.5m to 4.5m in the z-axis in the nuScenes coordinate system. This is our 4D volume $\mathcal{V}$, described in Sec. 3. We follow the state-of-the-art in point cloud forecasting and evaluate forecasting in a 1 second horizon and a 3 second horizon. We adopt the same setup as prior methods [18, 19]. On nuScenes, for 1s forecasting,, we take 2 frames of input and 2 frames of output at 2Hz; for 3s forecasting, we take 6 frames of input and 6 frames of output at 2Hz. For all other datasets, we always take 5 frames of input and 5 frames of output for both 1s and 3s forecasting.

Baselines First, we construct an aggregation-based ray-tracing baseline (similar to [11]). Specifically, we populate a binary occupancy grid given the aligned LiDAR point clouds from the past and present timesteps and use it for querying ground-truth rays. In addition to this, we compare our 4D occupancy forecasting approach to state-of-the-arts (SOTAs) in point cloud forecasting, including SPFNet [19] and S2Net [18] on the nuScenes dataset, and ST3DCNN [11] on the KITTI-Odometry dataset. For SPFNet [19] and S2Net [18], we are able to obtain the raw point cloud predictions from the authors and evaluate the results on the new metrics. For fair comparison, the S2Net results are based on a single sample from their VAE. For ST3DCNN [11], we retrain their models for 1s and 3s forecasting. In addition, the state-of-the-art approaches (barring ST3DCNN) tend to predict a confidence score for each point, indicating how valid the predicted point is; we evaluate the predicted point cloud both with and without confidence filtering, with a recommended confidence threshold at 0.05 [18, 19]. Quantitative and qualitative results with confidence filtering can be found in the supplement.

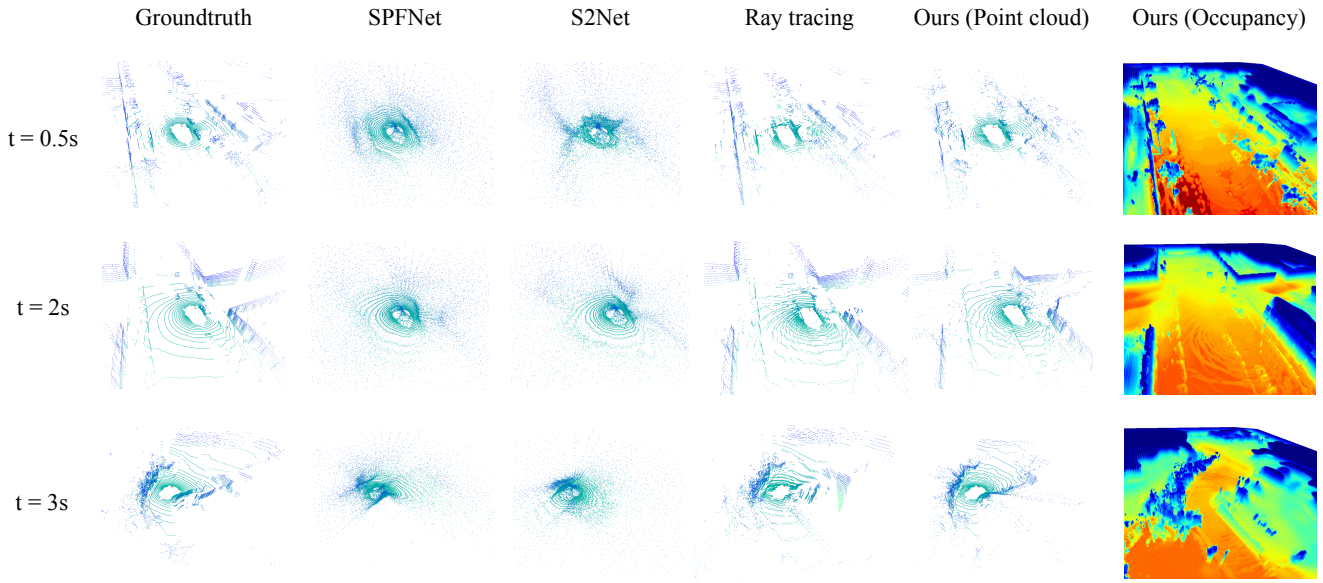

Figure 6. Qualitative results. We compare the point cloud forecasts of S2Net [18], SPFNet [19] and the raytracing baseline on the nuScenes dataset with our approach on three different sequences at different time horizon. Our forecasts look significantly crisper than the SOTA. This demonstrates the benefit of learning to forecast spacetime 4D occupancy with sensor intrinsics and extrinsics factored out. We also visualize the forecasted 4D occupancy at the corresponding future timestamp. As compared to simple *aggregation*-based raytracing, we are able to *spacetime-complete* 4D scenes. We highlight some potential applications in Fig. 7 and Fig. 8. We visualize a render of the predicted occupancy and the color encodes height along the z-axis.

Method	Horizon	L1 (m)	AbsRel (%)	Chamfer Distance (m^2) Near-field	Vanilla
S2Net [18]	1s	3.49	28.38	1.70	2.75
	3s	4.78	30.15	2.06	**3.47**
SPFNet [19]	1s	4.58	34.87	2.24	4.17
	3s	5.11	32.74	2.50	4.14
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 [4]. We see that the conclusions made from the proposed metrics are more in line with the qualitative results in Fig. 6. This 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).

Qualitative results on nuScenes We compare the forecasted point clouds from our 4D occupancy forecasting approach to SOTA on point cloud forecasting in Fig. 6, where we see a drastic difference in how the predicted point clouds look like. Our forecasts look significantly more representative of the scene geometry compared to SOTA. This demonstrates the benefit of learning to forecast spacetime 4D occupancy with sensor intrinsics and extrinsics factored out. Surprisingly, we find that aggregation-based raytracing is a competitive baseline, qualitatively better than the SOTA.

Method	Train set	Horizon	L1 (m)	AbsRel (%)	Chamfer Dist. (m^2) Near-field	Vanilla
ST3DCNN [11]	KITTI-O	1s	3.13	26.94	4.11	4.51
		3s	3.25	28.58	4.19	4.83
Ours	KITTI-O	1s	**1.12**	**9.09**	**0.51**	**0.61**
		3s	**1.45**	**12.23**	**0.96**	**1.50**
Ray tracing	-	1s	**1.50**	16.15	**0.62**	**0.76**
		3s	2.82	29.67	**4.01**	5.92
Ours	AV2	1s	1.71	**14.85**	2.52	3.18
		3s	**2.52**	**23.87**	4.83	**5.79**
Ours	KITTI-O$^{20\%}$	1s	1.25	9.69	1.95	2.27
		3s	1.70	14.09	4.09	5.09
Ours	AV2 + KITTI-O$^{20\%}$	1s	**1.19**	**9.30**	**0.54**	**0.64**
		3s	**1.67**	**13.40**	**1.24**	**1.80**

Table 2. Performance as a function of the available target dataset (in this case, KITTI-Odometry). With access to all of KITTI-O (**top**), our method outperforms the SOTA. With no access to KITTI-O (*i.e.* zero-shot sensor generalization in the **middle**), our method trained on AV2 outperforms the ray tracing baseline at 3s, though the baseline fares well at 1s. Note that both approaches still beat the SOTA [11] by a large margin. Finally, with access to only 20% of KITTI-O (**bottom**), our method fares quite well, particularly when trained on both AV2 and KITTI-O. Cross-dataset generalization and training is made possible by disentangling sensor intrinsics/extrinsics from scene motion.

However, in addition to this *aggregation*, our approach is also able to hallucinate or *spacetime-complete* both the future motion of dynamic objects and the occluded parts of the

static world. We also visualize the 3D forecasted occupancy at corresponding timestamps that our approach predicts "for free". Please refer to the caption for more details.

Results on nuScenes with new metrics We compare our 4D occupancy forecasting to SOTA on point cloud forecasting in terms of depth error along the future rays, following the evaluation protocol outlined in Sec. 4. We find that the 4D occupancy forecasting approach outperforms all baselines by significant margins in both 1s and 3s forecasting, reducing both the L1 and the absolute relative error by more than half, compared to the state-of-the-art methods on point cloud forecasting. The improvements here are consistent with the qualitative results in Fig. 6. As noted before, the raytracing baseline performs better than SOTA.

Results on nuScenes with old metrics We also evaluate by both vanilla (16) and near-field chamfer distance (17) following the protocol in Sec. 4. Our approach shines in terms of near-field chamfer distance. One contributing factor could be that our approach is specifically optimized for capturing occupancy evolution in the near field. In addition, S2Net [18] outperforms us in terms of vanilla chamfer distance, which is not surprising since we are incapable of deciding where rays end outside the predefined voxel grid.

Results on KITTI-Odometry Next, we use KITTI-Odometry to test our method in different settings with limited access to the target dataset. This mimics the setting where a next-generation sensor platform may be gradually integrated into fleet operations. Tab. 2 shows that with access to the full target dataset (KITTI-Odometry) for training, our method resoundingly outperforms the SOTA ST3DCNN [11]. Next, if no samples from the target dataset are available, one can employ either a non-learnable method such as our raytracing baseline, or one may pretrain on a (large) dataset with a different sensor platform. To this end, we find that our method trained on ArgoVerse2.0 outperforms the SOTA on KITTI-Odometry, while also outperforming raytracing baseline for long-horizon (3s) forecasting. Finally, with access to only 20% of KITTI-Odometry, our method pretrained on ArgoVerse2.0 and finetuned on KITTI-Odometry outperforms the alternatives. *To our knowledge, these are the first results in sensor transfer/generalization that illustrate the power of disentangling sensor extrinsics/intrinsics from scene motion.* Please see qualitative results in the supplement.

5.1. Architecture ablations

Here, we explore two other variants of our architecture: a *static* variant that predicts a single voxel grid for all future timesteps, and a *residual* variant that predicts a single static voxel grid with residual voxel grids for each output timestep. We evaluate these variants on nuScenes.

Arch.	Horizon	L1 (m)	AbsRel (%)	Chamfer Distance (m^2)	
				Near-field	Vanilla
S	1s	**1.28**	**9.27**	1.03	3.41
	3s	1.73	13.54	**1.40**	3.73
D	1s	1.40	10.37	1.41	**2.81**
	3s	**1.71**	**13.48**	**1.40**	4.31
S+R	1s	1.34	9.73	**1.00**	3.20
	3s	1.82	13.84	1.52	**3.54**

Table 3. We evaluate two variants of the proposed dynamic (D) architecture using the geometry forecasting metrics - static (S) and residual (S+R). We find that the static variant is a powerful baseline that beats our dynamic approach for 1s forecasting and by extension, the state-of-the-art.

The main observation is that the static variant is a powerful baseline for short-horizon forecasting. This is because a single voxel grid serves as a dense static map of the local region, and since an extremely high majority of the world remains static, this is expected to be a reasonable baseline for short-horizon forecasting. Note that this variant is still stronger than the ray tracing baseline in Tab. 1 because of its ability to hallucinate occluded parts of the world. On the other hand, the proposed *dynamic* variant (which predicts one voxel grid per future timestep), performs the best at long-horizon forecasting. With the residual variant, our hope was to separate dynamic scene elements from static regions, but in practice this decomposition fails as there is not enough regularization to force motion-based separation.

Since, the static variant outperforms the state-of-the-art on 1s forecasting, we analyse these variants further in the supplement by using the segmentation annotations on nuScenes-LiDARSeg [1] and computing the proposed metrics separately on foreground and background points. This helps us understand which regions in the scene contribute the least to the performance of these variants.

5.2. Applications

Generalization across sensors In Fig. 7 (captioned as new intrinsic-view synthesis), we show how one can render point clouds as if they are captured by different LiDAR sensors from the same predicted future occupancy. Typically, different LiDAR sensors exhibit different ray patterns when sensing. For the case shown, the nuScenes LiDAR is an "in-domain" sensor, i.e., the occupancy grid was predicted by a network learned over LiDAR sweeps captured by a nuScenes LiDAR. The KITTI and ArgoVerse LiDARs are "out-of-domain". We hope that learning of such a generic representation allows methods in sensor domain transfer [8, 21] to look at the task from the perspective of space-time 4D occupancy. The formulation we have laid out also

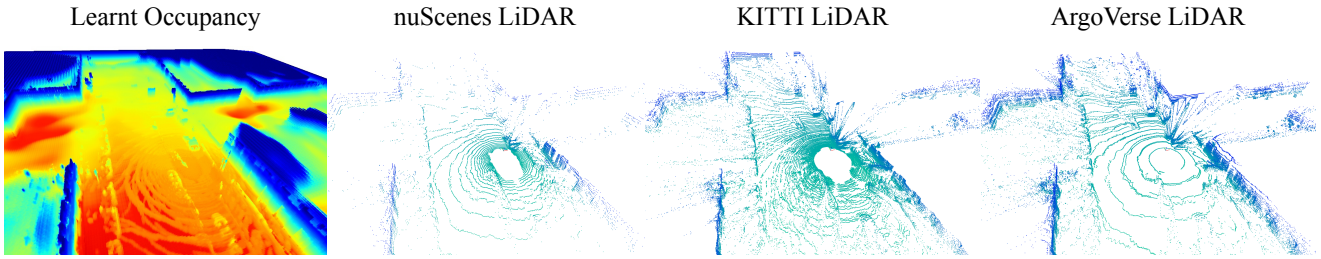

Figure 7. **Novel intrinsic-view synthesis** We show how to simulate different LiDAR ray patterns on top of the same learned occupancy grid. In this case, the future occupancy is predicted with historic LiDAR data scanned by nuScenes LiDAR (Velodyne HDL32E). First, we show the rendered point cloud under the native setting. Then, we show the rendered point cloud for KITTI LiDAR (Velodyne HDL64E, 2x as many beams). Finally, we have the rendered point cloud for Argoverse 2.0 LiDAR (2 VLP-32C stacked on top of each other). The fact that we can forecast occupancy on top of data captured by one type of sensor and use it to simulate future data for different sensors shows how generic the forecasted occupancy is as a representation. We support this generalization quantitatively in Tab. 2.

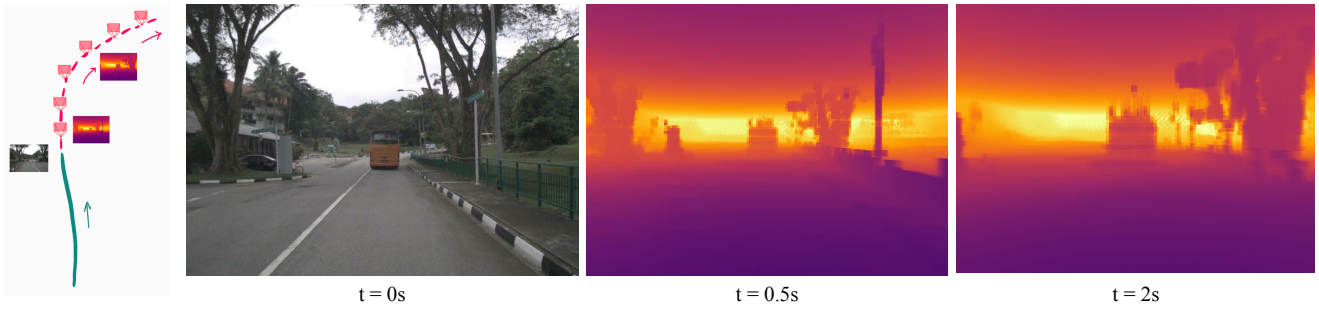

Figure 8. **Novel extrinsic-view synthesis** Dense depth maps rendered from the predicted future 4D occupancy from novel viewpoints. To render these depth maps, we take a novel future trajectory of the egovehicle. Placing the camera at each of these locations, always facing forward into the voxel grid (shown in the future dotted red trajectory on the left), gives us a camera coordinate system in which we can shoot rays from the camera center to every pixel in the image, and further beyond into the 4D occupancy volume. Every pixel represents the expected depth along its ray. The RGB image at $t = 0s$ is shown as reference and is not used in this rendering. For the depth maps, darker is closer, brighter is farther. Depth on sky regions is untrustworthy as no returns are received for this region from the LiDAR sensor.

makes it easy to train across different datasets, making zero-shot cross-dataset transfer possible for LiDARs [14, 15]. In the previous section and in Tab. 2, we highlight the first result in this direction, where our method trained on the ArgoVerse2.0 dataset when tested on KITTI-Odometry beats the prior art [11] on KITTI-Odometry. Furthermore, our proposed disentangling also allows for multi-dataset training, for which we point the readers to the supplement.

Novel view synthesis In Fig. 8 (captioned as new-extrinsic view synthesis), we show dense depth maps rendered from our learnt occupancy grid using novel egovehicle trajectories or viewpoints. Such dense depth of a scene is not possible to get from existing LiDAR sensors that return sparse observations of the world. Although classical depth completion [17] from sparse LiDAR input exists as a single-frame (current timestep) task, here we note that with our representation, it is possible to densify sparse LiDAR point clouds from the *future*, with such rendered depth maps backprojected into 3D. This dense 360° depth is evaluated on sparse points (with the help of future LiDAR returns) by our proposed ray-based evaluation metrics.

6. Conclusion

In this paper, we propose looking at point cloud forecasting through the lens of geometric occupancy forecasting, which is an emerging self-supervised task [7], originally set in the birds'-eye-view but extended to full 3D through this work. We advocate that this shift in viewpoint is necessary for two reasons. First, this shift helps algorithms focus on a generic intermediate representation of the world, i.e. its spatiotemporal 4D occupancy, which has great potential for downstream tasks. Second, this "renovates" how we formulate self-supervised LiDAR point cloud forecasting [11, 18, 19] by factoring out sensor extrinsics and intrinsics from the learning of shape and motion of different scene elements. In the end, we reiterate that the two tasks in discussion are surprisingly connected. We propose an evaluation protocol, that unifies the two worlds and focuses on a scalable evaluation for predicted geometry.

References

[1] nuScenes LiDARSeg. https://www.nuscenes.org/nuscenes#lidarseg. 7

[2] John Amanatides and Andrew Woo. A Fast Voxel Traversal Algorithm for Ray Tracing. In *EG 1987-Technical Papers*. Eurographics Association, 1987. 4

[3] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9297–9307, 2019. 5

[4] 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. 5, 6

[5] Sergio Casas, Abbas Sadat, and Raquel Urtasun. Mp3: A unified model to map, perceive, predict and plan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14403–14412, 2021. 2

[6] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *IJRR*, 32(11):1231–1237, 2013. 5

[7] 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. 2, 3, 8

[8] Ferdinand Langer, Andres Milioto, Alexandre Haag, Jens Behley, and Cyrill Stachniss. Domain transfer for semantic segmentation of lidar data using deep neural networks. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8263–8270. IEEE, 2020. 7

[9] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. 2019. 2

[10] Reza Mahjourian, Jinkyu Kim, Yuning Chai, Mingxing Tan, Ben Sapp, and Dragomir Anguelov. Occupancy flow fields for motion forecasting in autonomous driving. *IEEE Robotics and Automation Letters*, 7(2):5639–5646, 2022. 2

[11] 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, 2, 5, 6, 7, 8

[12] 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. 2, 3

[13] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3504–3515, 2020. 2

[14] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12179–12188, 2021. 8

[15] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE transactions on pattern analysis and machine intelligence*, 2020. 8

[16] Abbas Sadat, Sergio Casas, Mengye Ren, Xinyu Wu, Pranaab Dhawan, and Raquel Urtasun. Perceive, predict, and plan: Safe motion planning through interpretable semantic representations. *European Conference on Computer Vision*, 2020. 2

[17] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. Sparsity invariant cnns. In *International Conference on 3D Vision (3DV)*, 2017. 8

[18] 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, 5, 6, 7, 8

[19] 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, 5, 6, 8

[20] 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. 1, 2, 5

[21] Li Yi, Boqing Gong, and Thomas Funkhouser. Complete & label: A domain adaptation approach to semantic segmentation of lidar point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15363–15373, 2021. 7