Self-Supervised Sparse Sensor Fusion for Long Range Perception (Supplementary Information)

Edoardo Palladin*[1] Samuel Brucker*[1]
Filippo Ghilotti[1] Praveen Narayanan[1] Mario Bijelic[1,2] Felix Heide[1,2]
[1]Torc Robotics [2]Princeton University *Equal contribution
https://light.princeton.edu/LRS4Fusion

The following document provides supplemental materials to support the findings in the main manuscript. In Section 1, we report additional details about the experimental long-range highway dataset, and we include samples extracted from the recordings. In Section 2, we provide additional implementation information on the components of the proposed architecture. In Section 3, we describe the experimental setup for the Depth, Lidar Forecasting and Object detection experiments. In Section 4, we detail the implementation and training of baseline methods in the different tasks presented in the main manuscript. In Section 5, we present additional quantitative results for object detection and a modality ablation study for LiDAR forecasting. Finally, in Section 6, we show additional qualitative results.

Contents

Figure 1. **Long Range Dataset Image Samples.** We present a visualizations of some of the 60000 image samples part of our high-resolutions (3848 × 2168 pixels) cameras, in different environments, light and weather conditions.

1. Long-Range Experimental Highway Dataset

Dataset Description. We introduce a dataset designed for long-haul trucking, with a focus on long-range perception critical for high-speed highway environments. Unlike popular public datasets, like NuScenes [2], Argoverse 2 [20], KITTI [5], and the Waymo Open Dataset [18], which primarily rely on LiDAR sensors capable of capturing data in the short- or mid-range, our dataset employs 4D LiDAR sensor (AEVA AERIS II), which extend detection range up to 400 meters and directly measure radial velocity for each point through Frequency-Modulated Continuous-Wave (FMCW) technology. Figure 2 compares the distribution of labeled bounding boxes by distance from the ego vehicle in our dataset, NuScenes [2], and Argoverse 2 [20]. The comparison highlights that our dataset addresses a critical gap in public benchmarks by providing long range scenarios necessary for training and evaluating autonomous driving performance at extended distances, which is essential for safe and reliable highway driving.

Mounted atop a semi-truck's driver cabin within a dedicated sensor module, our system integrates both LiDAR and camera data to enhance object detection and depth prediction capabilities. The imaging component uses OnSemi AR0820 cameras with 1/2-inch CMOS sensors, capturing raw RCCB data at a high resolution of 3848 × 2168 pixels and nearly complete 360° coverage at a frame rate of 5 Hz. High-resolution images aid in capturing more details and boundaries, which is especially important for perception at long ranges where smaller objects and subtle variations in texture or color can hinder model capabilities. Assessing the long-range camera with a horizontal FoV of 30°, the camera has a range of up to $\approx$ 1070m assuming a sedan with 1.5m height and 100 pixels for reliable detection. In Figure 1 we present some example samples extracted from our data capture system, in different light-road conditions and scenarios. Both the camera and LiDAR sensors are synchronized to ensure cohesive data acquisition, with manual calibration performed between recordings to maintain precision. To ensure accurate annotations, the dataset leverages both camera and LiDAR data in a complementary manner. The dataset comprises 60,000 unlabeled frames and 36,000 frames with manual annotations realized leveraging both camera and LiDAR data in a complementary manner, spanning seven object categories: Bike, Passenger-Car, Person, RoadObstruction, SemiTruck-Cab, SemiTruck-Trailer, and Vehicle.

FMCW Velocity Capturing. FMCW LiDAR measures both distance and radial velocity by transmitting a continuous wave whose frequency is modulated over time. The transmitted signal can be expressed as

$$s_{\text{tx}}(t) = A\cos\big(2\pi\big(f_0 + \mu t\big)t\big). \tag{1}$$

When the signal reflects off a moving target at range R and radial velocity v_r, it is delayed by $\tau = \frac{2R}{c}$ and undergoes a Doppler shift $f_D = \frac{2v}{\lambda}$. The received signal becomes

$$s_{\text{rx}}(t) = A\cos\Big(2\pi\big[f_0 + \mu(t-\tau) + f_D\big](t-\tau)\Big). \tag{2}$$

Mixing the transmitted and received signals a so called beat frequency can be obtained$f_{\text{beat}} \approx \mu\tau - f_D$ from which both range and velocity can be determined. It is important to note that the radial velocity v_r is defined as the component of an object's velocity v along the line of sight and can, therefore, be obtained as

$$v_r = v\cos\theta, \tag{3}$$

where θ is the angle between the object's velocity vector and the line-of-sight. For an object moving perpendicularly to the line of site, it´s true that $\cos\theta = 0$, shwoing that for perpendicular motion, the radial velocity is zero. Since our LiDAR system offers complet 360° coverage, and objects have a non-zero spatial extent, points will exhibit zero velocity if belonging to vehicles driving on a circular trajectory around the ego one.

2. Implementation Details

We provide implementation details, including the model architecture, downstream task setup, pre-training procedures, and hyperparameters.

2.1. Model Architecture

The model architecture includes the input sensors, depth prediction network, sparse convolutional operations, sparse windowed attention, and the occupancy and velocity decoders.

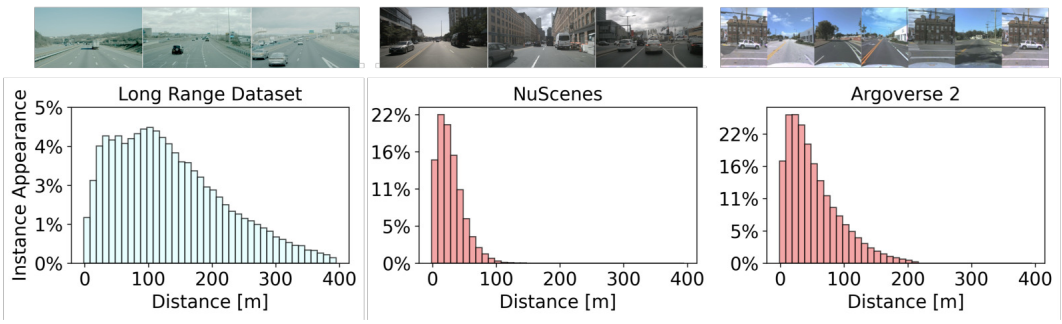

Figure 2. Comparison of labeled bounding box distributions over distance from the ego vehicle in our Long Range Dataset, NuScenes [2], and Argoverse 2 [20].

2.1.1. Input Sensors.

- **Images**: We make use of 5 cameras, two front/side wide lens cameras, 1 narrow camera in the front for long range perception and 2 rearward facing cameras. We downscale the input images to a resolution of 544 x 960.
- **LiDAR**: We make use of a single joint point cloud composed of calibrated multiple scans recorded by the different LiDAR sensors. We clip the input LiDAR point cloud to $[-100 : +250]$ on the x axis, $[-100 : +100]$ and the y axis and $[-12 : +12]$ on the z axis and voxelize it with voxels of dimension $0.20m$ in the x and y axis and 0.38 on the z axis.

2.1.2. Depth Network.

In this section, we describe the implementation details of our depth completion model, expanding upon the methodology outlined in the main paper.

Multi-Scale Iterative Refinement. The model iteratively refines depth predictions across multiple spatial scales. At each scale, the depth-backbone extracts contextual features $F = \{F_1^i, F_2^i, F_3^i, F_4^i\}$ and corresponding confidence maps used to guide depth estimation. Each scale's feature map is processed independently by the depth-backbone network:

$$(h_t, C_{\text{inp}}) = \mathcal{B}(F), \tag{4}$$

where h_t is the hidden state, and C_{inp} represents the decoded confidence features derived from input features F.

Minimal Gated Unit (MGU) Update Block. To efficiently update depth predictions across iterations, we employ a Minimal Gated Unit (MGU), simplifying traditional gated architectures by combining reset and update gates. Each iterative refinement step follows

$$h_t = (1 - r_t) \odot h_{t-1} + r_t \odot \phi\left(\mathcal{C}_h([h_{t-1}, x_t])\right), \tag{5}$$

with gate computation

$$r_t = \sigma\left(\mathcal{C}_r([h_{t-1}, x_t])\right), \tag{6}$$

where $\mathcal{C}_h$ and $\mathcal{C}_r$ are convolutional layers, and x_t is the concatenation of context features and gradient discrepancies. Additionally, the MGU module employs separate convolutional gates consisting of convolutional layers followed by batch normalization and nonlinear activations. This gated convolutional approach effectively integrates image-derived context and depth gradient discrepancies, balancing model simplicity and predictive accuracy.

Depth Integration Module. The depth integration module updates the depth estimate iteratively

$$d_{t+1} = d_t - \Delta d, \tag{7}$$

with

$$\Delta d = f_{\text{update}}\left(\nabla d_t - g, (d_t - s_d) \odot M, C_{dg}, C_{\text{inp}}\right), \tag{8}$$

where Δd is computed via convolutional layers in f_{update}, taking gradient discrepancies, depth discrepancies masked by LiDAR validity (M), and confidence maps (C_{dg}, C_{inp}) as inputs. The depth integration module employs three sequential convolutional layers with ReLU activations, transforming input discrepancies and confidence maps into incremental depth corrections. This iterative convolutional refinement ensures depth updates balance fidelity to sparse measurements and smoothness informed by image features.

2.1.3. Sparse Operations.

For the sparse operations we rely on the SPConv implementation [3]. For the sampling operation in the sparse voxels, used to lift camera features in 3D, to sample features in the occupancy and velocity decoders and to excecute sparse windowed attention, we implement a custom ball query interpolation operations, that exploit the integer nature of the coordinates representation of occupied voxels, removing the need for KNN and rely solely on indexing operations. The dimension of the hidden voxels in the latent features at scale 1 matches the input voxel size ($0.20m$ in the x and y axis and $0.38m$ on the y axis) and doubles in every subsequential scale, with shape $1.60m$ in the x and y axis and $3.00m$ on the z axis in the lowest scale.

2.1.4. Sparse Windowed Attention.

For the Sparse Windowed Attention, we set a 3D local window of 3x3x3, effectively attending 1 occupied voxel in the current frame to 27 voxels in the past frame. For the un-occupied voxels inside the sampling window we set an attention mask to exclude them from the attention mechanism. Sampled features are first feed into a projection module composed by 2 linear layers, with a layer norm and a Gelu activation, then a positional embedding is added encoding the 3D position of the feature and temporal difference between the two fused frames. We implement the attention mechanism as a multi-head attention, with 4 heads and latent dimension of 16. The resulting features are passed to a last projection module.

2.1.5. Occupancy and Velocity Decoders.

Occupancy and Velocity heads are implemented as semi-implicit decoders. They take a query position and time as input and they predict the occupancy and velocity values. The two heads shares the sampling component over the sparse grid, but the sampled features are passed to two independent branches to predict occupancy and velocity. The query position is converted from real world coordinate in LiDAR frame, to sparse volume coordinates; we then sample a first time in the sparse volume at scale 2, considering a window composed by the point itself and the 6 direct neighbors. The sampled features are then passed into a projection layers and summed with an embedding produced by a linear projection of the query position and time to generate the sampling offset. The offset is now summed to the original query position to generate a refined sampling point. We sample again from the sparse volume, this time from all 4 hidden volume scales, taking 1 neighbor features for the 2 smallest scale and 7 from the biggest 2. We concatenate the first sampled feature with the latest 4 features and pass the resulting tensor to two independent velocity and occupancy heads. These are parametrized as a series of 3 residual blocks, where in the first block we add the query position and time embedding. The occupancy head presents an additional sigmoid activation at the end to produce occupancy values in the $0:1$ range. To visualize occupancies, we apply an threshold of 0.5, displaying regions where the predicted occupancy exceeds this value while omitting lower values.

The anticipation of dynamic motion for future forecasting in our method relies on three key design choices. (i) We utilize multi-scale interpolated voxels, enabling a large receptive field that captures inter-voxel motion across spatial scales. (ii) Sparse temporal fusion integrates past observations into the current encoding, effectively embedding historical actor motion to inform predictions. (iii) Additionally, when available, 4D LiDAR data containing velocity measurements is explicitly encoded, providing direct velocity cues that further enhance the model's ability to anticipate future positions.

2.2. Down-Stream Tasks

LiDAR Reconstruction To facilitate the evaluation of our pre-training scheme, we additionally reconstruct the recorded LiDAR point clouds at different timestamps. To this end, we generate a query ray for each point in the current and future point clouds to reconstruct the distance at which the LiDAR beam intersects a surface. We first sample uniformly $Q_{\text{LiDAR\_rec}}$ points along the ray and then query the model occupancy decoder head from Sec. 2.1.5 at every sampling position. The generated occupancy predictions are then concatenated and processed with the LiDAR reconstruction decoder $f_{\text{LiDAR\_rec}}$, composed by a sequence of 1D convolutions and linear layers. The final linear layer predicts a scalar value per ray, representing the ray length.

$$\hat{d}_r = f_{\text{LiDAR\_rec}}(\hat{\sigma}_{\mathbf{r}}) \tag{9}$$

where $\hat{\sigma}_{\mathbf{r}} \in \mathbf{R}^{N,4}$ and $\hat{\sigma}_{\mathbf{r}}^{\mathbf{i}} = f_{\text{dens}}(Q^i_{\text{LiDAR\_rec}})$ with $i = 0, ...N$. We supervise this depth measurement against the ground truth d_r with an ℓ_1 loss. By prompting the model with denser query rays, sampled according to the camera's intrinsics (including focal length and principal point), the reconstruction head can generate dense depth maps that align with the perspective geometry of any input view.

Object Detection To evaluate the proposed architecture on the OD task we replace the pre-training occupancy reconstruction head with an object detection head. Due to the modularity of the method, any OD head can be used. In the experiments

Table 1. Network Details of LRS4Fusion.

Component	Sub-Component	Layers	Parameters
Image Branch	Backbone	Vim - Seg	embed dim: 192, depth: 24, patch size: 16
	Neck	FPN for OD	in: [192, 192, 192, 192], out: 64
Depth Model	Depth Backbone $\mathcal{B}$	ctx dec1 - Conv2D	(64, 64), k=3, s=1, p=1, BatchNorm, ReLU
		ctx dec0 - Conv2D	(64, 64), k=3, s=1, p=1
		cfi dec1 - Conv2D	(32), k=3, s=1, p=1, BatchNorm, ReLU
		cfi dec0 - Conv2D	(32, 1), k=3, s=1, p=1, Sigmoid
	BasicUpdateBlockMGU $\times 6$	Encoder - Conv2D-d	(1, 64), k=7, p=3, ReLU
		Encoder - Conv2D-d	(64, 32), k=3, p=1, ReLU
		Encoder - Conv2D-g	(2, 64), k=7, p=3, ReLU
		Encoder - Conv2D-g	(64, 32), k=3, p=1, ReLU
		Encoder - Conv2D	(64, 61), k=3, p=1, ReLU
		MGU - Conv2d-z	(128, 64), k=3, p=1, BatchNorm, TanH
		MGU - Conv2d-n	(128, 64), k=3, p=1, BatchNorm, TanH
		DepthGradHead - Conv2d	(64, 128, 2), k=3, p=1, ReLU
		ConfidenceHead - Conv2d	(64, 32, 2), k=3, p=1, ReLU, Sigmoid
	DepthIntegrationModule	IntegrationUpdateBlock - Conv2D	(6, 64, 54, 54, 54, 1), ReLU
	ImageDecoder	ConvReduce - Conv2D	(64, 32), k=3, s=1, p=1, RelU
		Deconv - ConvTrans2D	(32, 32, 32), k=4, s=2, p=1, RelU
		ConvOut - Conv2D	(32, 3), k=3, s=1, p=1, Sigmoid
LiDAR Branch	Backbone - Unet	Encoder	((32,), (64, 64,), (128, 128,), (128, 128, 128), (256, 256, 256))
	Backbone - Unet	Decoder	((256, 256, 128), (128, 128, 128), (128, 128, 64), (64, 64, 32), (32, 32, 32))
Sparse MM Fusion	Sparse Conv	SubMConv3d	(96, 64), LayerNorm, ReLU
Late Encoder	Scale 1	complBlock	(32, 32), k=3, Batch Norm, only xy=True
		aggreBlock	(32, 32), k=3, Batch Norm, pooling: True
	Scale 2	complBlock	(32, 32), k=3, Batch Norm, only xy=False
		aggreBlock	(32, 64), k=3, Batch Norm, pooling: True
	Scale 3	complBlock	(64, 64), k=3, Batch Norm, only xy=True
		aggreBlock2D	(64, 128), k=5, Batch Norm, pooling: True
	Scale 4	complBlock	(128, 128), k=3, Batch Norm, only xy=True
		aggreBlock2D	(128, 256), k=5, Batch Norm, pooling: False
Temporal Fusion	query proj	MLP	(32, 16, 16), Layer Norm, GeLU
	key proj	MLP	(64, 16, 16), Layer Norm, GeLU
	out proj	MLP	(16, 16, 32), Layer Norm, GeLU
	pos emb query	MLP	(4, 16, 16), Layer Norm, GeLU
	pos emb key	MLP	(4, 16, 16), Layer Norm, GeLU
	Attnention	MultiHead	Heads: 4, embed dim: 16
Occupancy Head	Query Proj	Linear	(4, 16)
	Feat Proj	Linear	(64, 16)
	Offset 1	Linear	(16, 16)
	Offset 2	Linear	(16, 16)
	Offset Head	Linear	(16, 3)
	Query Proj Block 1	Linear	(4, 16)
	Feat Proj 1	Linear	(544, 16)
	Residual1	MLP	(16, 16, 16), ReLU
	Feat Proj 2	Linear	(544, 16)
	Residual2	MLP	(16, 16, 16), ReLU
	Feat Proj 3	Linear	(544, 16)
	Residual3	MLP	(16, 16, 16), ReLU
	Occ Head	MLP	(16,1), Sigmoid
Occupancy Head	Feat Proj 1	Linear	(544, 16)
	Residual1	MLP	(16, 16, 16), ReLU
	Feat Proj 2	Linear	(544, 16)
	Residual2	MLP	(16, 16, 16), ReLU
	Feat Proj 3	Linear	(544, 16)
	Residual3	MLP	(16, 16, 16), ReLU
	Vel Head	MLP	(16,1)
LiDAR Head	Occ Encoder	Linear	(1, 128)
	Depth Ray Encoding	Linear	(1, 128)
	CNN Pooling	Conv1D	(128, 64, 32, 16, 16, 16, 8) K=4, S=2, ReLU
	MLP Decoder	Linear	(232, 64, 32, 16, 1), ReLU
Object Detection	Backbone	SECOND	in: 256, out: [128, 256], layer nums: [5, 5], layer strides: [1, 2], Batch Norm
	Neck	SECONDFPN	in: [128, 256], out: [256, 256], upsample strides: [1, 2], Batch Norm
	Head	CenterHead	in: [256, 256], conv ch: 64, single head

conducted we first apply a max pooling on the pillar axis over the smaller scale 3D sparse features, followed by a densification and flattening operation to bring the features into a BEV grid. We employ two common object detection losses:

- L1 bounding boxes regression loss $\mathcal{L}_{L1BB}$

- Gaussian Focal Loss $\mathcal{L}_{Gaussian}$ [10, 26]

The total object detection loss is then determined by

$$L_{OD} = w_{L1BB} * \mathcal{L}_{L1BB} + w_{Gaussian} * \mathcal{L}_{Gaussian} \tag{10}$$

with $w_{L1BB} = 0.25$ and $w_{Gaussian} = 1$.

2.3. Pre-Training Details

Image and Depth Pre-Training. Our pre-training consists of two stages. First, we train the image feature encoder and depth prediction components using three primary losses: image reconstruction, depth supervision, and feature distillation. Specifically, the losses include:

- **Image Reconstruction Loss** $\mathcal{L}_{\text{img}}$: A weighted combination of the structural similarity (SSIM) loss [19] and the $\mathcal{L}_1$ loss between the input and reconstructed images: $\mathcal{L}_{\text{img}} = 0.7 \cdot \text{SSIM} + 0.3 \cdot \mathcal{L}_1$. This supervision prevents the model from focusing solely on depth prediction and helps retain semantic information. The loss is scaled by a factor of 40.
- **Depth Supervision Loss:** The model jointly optimizes multiple depth-related losses:
  - **LiDAR Loss** $\mathcal{L}_{\text{lidar}}$: A weighted L1-L2 (BerHu [8]) loss applied to sparse depth measurements.
  - **Gradient Consistency Loss** $\mathcal{L}_{\text{seqgrad}}$: Ensures the learned depth gradient aligns with computed gradients from the current depth map [28].
  - **Smoothness Loss** $\mathcal{L}_{\text{smooth}}$: Encourages smoothness using an edge-aware term based on image gradients.
  - **Monocular Depth Loss** $\mathcal{L}_{\text{mono}}$: Aligns predicted depths to a pretrained monocular depth estimator [24], using RANSAC regression with LiDAR depth to mitigate scale and shift discrepancies.

  These loss terms are weighted and summed, with iterative predictions contributing proportionally decreasing weights to encourage coarse-to-fine refinement:

$$\mathcal{L}_{\text{depth}} = 3\mathcal{L}_{\text{lidar}} + 5\mathcal{L}_{\text{seqgrad}} + 0.25\mathcal{L}_{\text{mono}} + \mathcal{L}_{\text{smooth}}. \tag{11}$$

- **Feature Distillation Loss** $\mathcal{L}_{\text{feat}}$: A combination of L1 loss and cosine embedding loss between extracted DINOv2 [14] features and Vim [27] features projected using a small adapter head, following [16]. This loss is weighted by a factor of 5.

The total loss for stage one training is:

$$\mathcal{L}_{\text{stage1}} = \mathcal{L}_{\text{img}} + \mathcal{L}_{\text{depth}} + \mathcal{L}_{\text{feat}}. \tag{12}$$

Figure 3 illustrates the stage one training process.

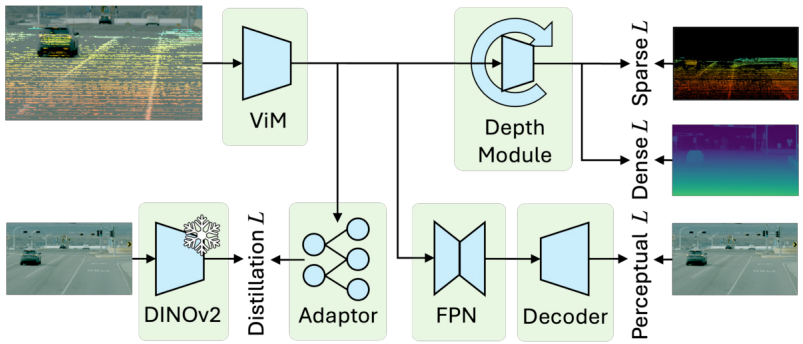

Figure 3. Overview of image Feature Extraction and Depth Prediction training. Image and project LiDAR scans are encoded with Vim [27] backbone. The resulting features are passed to a feature pyramid (FPN) and then are passed to the depth model to predict dense depth. Vim Features are supervised with a distillation loss to DINOv2 [14] features; FPN features are supervised to reconstruct the original image with a light weight decoder; depth is supervised with LiDAR depth and monocular depth where LiDAR scans cannot be projected into the image frame.

Model Training. In the second stage, we train the full model by supervising the past, current, and predicted future frames with occupancy and velocity prediction. The total loss is here given by

$$\mathcal{L}_{\text{stage2}} = \mathcal{L}_{\text{occ}} + \mathcal{L}_{\text{velo}} \tag{13}$$

Table 2. Training Hyper-Parameters.

Dataset	Stage	Iterations	LR
NuScenes	Stage 1	$100 * 10^3$	Constant $4 * 10^{-4}$
	Stage 2	$100 * 10^3$	Constant $4 * 10^{-4}$
	Lidar Forecasting	$40 * 10^3$	Cyclic $2 * 10^{-3}$
LongRange	Stage 1	$20 * 10^3$	Constant $4 * 10^{-4}$
	Stage 2	$40 * 10^3$	Constant $4 * 10^{-4}$
	Lidar Forecasting	$20 * 10^3$	Cyclic $2 * 10^{-3}$
	Object Detection	$40 * 10^3$	Cyclic $1 * 10^{-3}$

For Long Range Dataset we weight the $\mathcal{L}_{\text{occ}}$ by 10 and $\mathcal{L}_{\text{velo}}$ by 1. For NuScenes Dateset, giving the abscence of velocity supervision we turn off $\mathcal{L}_{\text{velo}}$.

For the last and third stage to learn to predict object detections we supervise the method applying an Object Detection loss $\mathcal{L}_{\text{detection}}$ following [23].

Occupancy Loss ($\mathcal{L}_{\text{occ}}$)**.** The occupancy is supervised using occupancy ground-truth generated from the LiDAR data. We treat a LiDAR ray as unoccupied up to the point where the LiDAR detects an object. After this point, we follow [1] and account for a small margin δ that is marked as occupied. We set $\delta = 0.1$. We generate N query points, sampling an equal number of occupied and unoccupied points, $N_O = N_U$, from the ground truth. For each query point, we predict an occupancy value $\hat{o}$ and compare it to the binary occupancy ground truth o_{gt}. We use the binary cross-entropy loss defined as:

$$\mathcal{L}_{\text{density}} = -\frac{1}{N}\left(\sum_{q\in Q^-}\log(1-\hat{o}_q) + \sum_{q\in Q^+}\log(\hat{o}_q)\right) \tag{14}$$

Where N is the total number of sampled points, $\hat{o}_q$ is the predicted occupancy for point q, and $o_{\text{gt},q}$ is the ground truth density for that point.

Velocity Loss ($\mathcal{L}_{\text{velo}}$)**.** The velocity loss supervises the predicted velocity using the LiDAR-derived ground truth. For unoccupied regions, we assign $v_{\text{gt}} = 0$, and for occupied regions, we use the velocity measured by the 4D LiDAR, $v_{\text{gt}} = v_L$. The velocity loss is defined as

$$\mathcal{L}_{\text{velo}} = \sum_{q=1}^{N} \left|\hat{v}_q - v_{\text{gt},q}\right|, \tag{15}$$

where $\hat{v}_q$ and $v_{\text{gt},q}$ are the predicted and the ground truth velocity for point q.

2.4. Hyper-Parameters

For all the trainings we use AdamW as optimizer with 0.1 weight decay and 35 gradient clipping. We train stage 1 for $20*10^3$ iterations with batch size 8 with a constant learning rate of $4 * 10^{-4}$ on the Long Range Dataset and for $100 * 10^3$ on the NuScenes dataset. We train stage 2 for $40*10^3$ with a constant learning rate of 10^{-4} and warm-up of $1*10^3$ iterations on the Long Range Dataset and for $100 * 10^3$ on the NuScenes dataset. For the LiDAR forecasting task we freeze the entire model and train only the LiDAR reconstruction head for $20 * 10^3$ iterations on the Long Range Dataset and for $40 * 10^3$ epochs on NuScenes with a cyclic learning rate with a peak of 0.002 (Tab. 2). For the object detection task we train the full model for $40 * 10^3$ iterations with a cyclic learning rate with a peak of 0.001.

Object Detection Implementation For Object Detection we use a Second FPN neck and a CenterPoint head [23].

3. Experimental Setup

Following we describe the experimental setup of depth prediction and the downstream tasks LiDAR forecasting and object detection.

3.1. Depth Prediction

Accumulated LiDAR Ground Truth. We evaluate depth prediction using an accumulated LiDAR ground truth, rather then a sparse, single-frame one, to achieve a denser and more consistent reference.

To generate this ground-truth we adapt a SLAM-based algorithm [17] to our unique sensor setup, and accumulate 150 LiDAR scans from each scene. Since our highway dataset is rich of dynamic actors, we pre-process each LiDAR recording by removing points identified as moving, reducing the number of floaters and artifacts in each map.

The moving-objects segmentation leverages the sensor's radial velocity measurements: we split positive and negative velocity components, compute their respective means (μ^+),(μ^-) and standard deviation (σ^+),(σ^-) and classify as moving if its velocity exceeds $\mu^+ + 2.0\sigma^+$ for positive velocities and $|\mu^-| + 2.0\sigma^-$ for negative velocities. To address cases in which an object moves perpendicularly to the sensor, thus showing minimal radial velocity, we further rely on state of the art moving object segmentation method [13] to complement the prediction.

In order to re-generate single frames for validation, for each timestamp t we exploit the pose provided by the slam algorithm to transform the accumulated points back into the original LiDAR coordinates, reintroducing moving points removed in the pre-processing relevant to the current t.

We then iterate at 2 levels to remove occluded points not visible from the current viewpoint. First, at LiDAR level we employ a simplified binned ray-tracing strategy: we subdivide the space in bins (i, j) and record the minimum range $r_{\text{min}}^{(i,j)}$ of points in each bin. We define a range-dependent threshold function $T(r) = 1 + \alpha r$ and retain any point k in bin (i, j) if

$$r_k \leq r_{\text{min}}^{(i,j)} + T(r_k) \tag{16}$$

and otherwise discard it as occluded. Second, at image level, we project the remaining LiDAR points into the camera frame and we compare each valid depth value d at pixel (x, y) against $\min(D_{\text{local}})$, the minimum depth in a local circular neighborhood of radius $r = 3$. If

$$\frac{d}{\min(D_{\text{local}})} > \tau, \quad \tau = 1.25 \tag{17}$$

then d is considered occluded and excluded from further consideration. Otherwise, the new depth value updates the buffer in that local region. Iterating over all pixels, in ascending depth order, ensures that only non-occluded points remain in the final depth map.

This refined, dense accumulated LiDAR serves as a robust, dense ground-truth for evaluating depth models only: the full model remains independent of this pre-processing step and can be trained without it.

In Figure 4 we show an example of our accumulated LiDAR scans compared to the sparse version, showing consistency in occlusions between the two as well as higher density at all ranges.

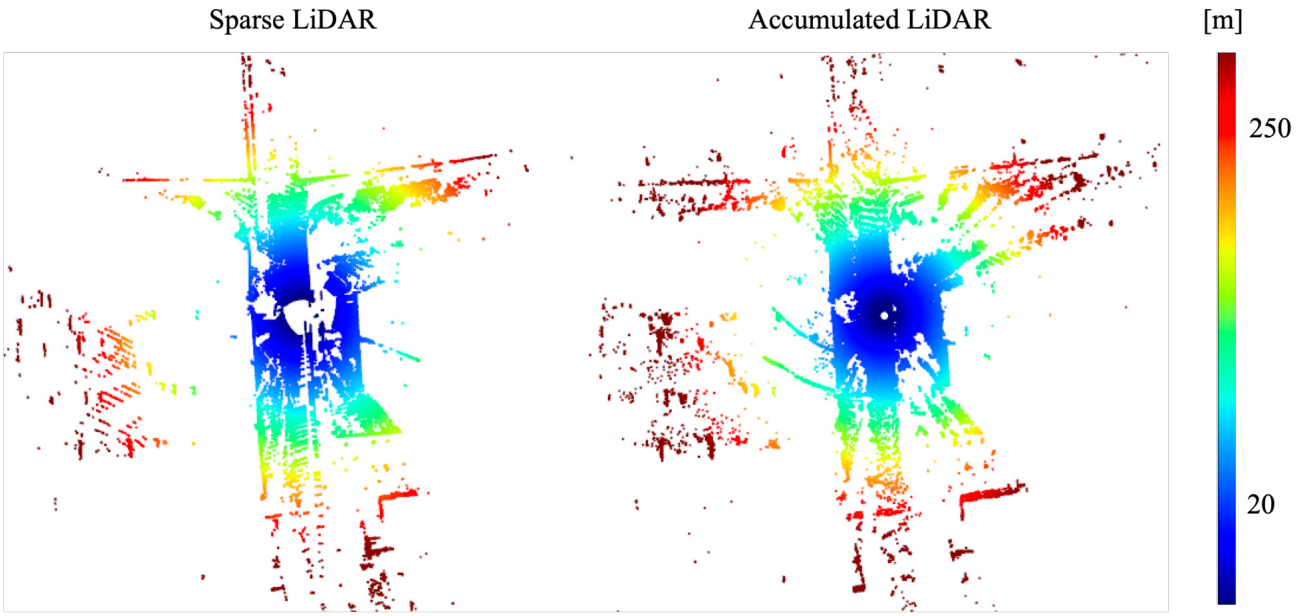

Figure 4. **Accumulated Evaluation LiDAR Ground-Truth.** We show an example of our accumulated LiDAR scan (right), retaining 3× the number of points compared to the normal sparse version (left), with increased ranges and accurate occlusion management.

3.2. LiDAR Forecasting.

For the LiDAR forecasting tasks we allign to prior work [22] and computer Chamfer Distance (CD) between predicted and ground truth point clouds. For NuScenes dataset we evaluate CD inside the ROI $[-51.2m : +51.2m]$ on the x axis, $[-51.2m : +51.2m]$ on the y axis and $[-51.2m : +51.2m]$ on the z axis. For the LongRange dataset we evaluate CD inside the ROI $[-100m : +250m]$ on the x axis, $[-100m : +100m]$ on the y axis and $[-12m : +12m]$ on the z axis. We calculate the Chamfer distance as

$$CD = \frac{1}{2N} \sum_{\mathbf{x} \in \mathbf{X}} \min_{\hat{\mathbf{x}} \in \hat{\mathbf{X}}} \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 + \frac{1}{2M} \sum_{\hat{\mathbf{x}} \in \hat{\mathbf{X}}} \min_{\mathbf{x} \in \mathbf{X}} \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 \tag{18}$$

where $\mathbf{x} \in \mathbf{R}^{Nx3}, \hat{\mathbf{x}} \in \mathbf{R}^{Mx3}$ represent the ground-truth and predicted point clouds, respectively.

For the Long Range Experimental Dataset we additionally report L1 error between forecasted depths along each LiDAR ray $\hat{\mathbf{D}} \in \mathbf{R}^N$ and ground truth depths $\mathbf{D} \in \mathbf{R}^M$ as

$$L1 = mean(abs(\mathbf{D} - \hat{\mathbf{D}})) \tag{19}$$

3.3. Object Detection.

To assess the performance on Object Detection task, we rely on standard 3D object detection metrics. These include Mean Average Precision (mAP) and NuScenes Detection (ND) Score, which combines mAP with other auxiliary metrics (mATE, mASE, mAOE, mAVE, and mAAE) to provide a comprehensive evaluation of 3D object detection performance, capturing the precision, accuracy in positioning, scale estimation, and orientation of detected objects in 3D space. Metrics are calculated for detections inside the considered ROI of $[-100m : +250m]$ on the x axis, $[-100m : +100m]$ on the y axis, with true positives defined by a distance threshold of less than 6 m.

4. Baseline Implementation Details

In the following we provide training and implementation details for depth estimation, LiDAR forecasting and object detection.

4.1. Depth Prediction

Completion-Former and OGNI-DC. For Completion-Former [25] OGNI-DC [28] we rely on the provided implementation and training code. We initialize both with KITTI pretrained weights and finetune the models for 10 epochs on our training split.

4.2. LiDAR Forecasting

4D-Occ. For 4D-Occ [7] results we rely on the codebase that was provided with the the publication. The model is extended to the full ROI of the Long Range Dataset ($-100m : +100m$ on y axis, $-100m : +250m$ on x axis) and the voxels dimension are set to $0.5m$ due to GPU memory limitations. For the $1s$ setting we take as input a queue of length 5 at $5Hz$ and for the $3s$ setting we keep the queue length to 5 but we downsample the data frequency by 3 times. The model is trained for 15 epochs with the original scheduling.

ViDAR. For ViDAR [22] results also builts ontop of the publicly available implementation by the original authors. The model is extended to the full ROI of the Long Range Dataset ($-100m : +100m$ on y axis, $-100m : +250m$ on x axis) but the BEV grid size is kept to the original proposed configuration ($200x200$) due to GPU memory limitations. For the $1s$ setting we take as input a queue of length 5 at $5Hz$ and for the $3s$ setting we keep the queue length to 5 but we downsample the data frequency by 3 times. The model is pre-trained on the NuScenes checkpoint and fine-tuned for 15 epochs on the Long Range Dataset. In Tab. 3 we report a small ablation study on ViDAR performance with different ROIs.

4.3. Object Detection

In this section we provide Implementation and training details for object detection baseline models. For all training we use AdamW scheduler, gradient clipping set to 35 and 0.1 weight decay.

PointPillars. For PointPillars [9] results, we train for 20 epochs with a cyclic learning rate starting at 0.0005 with peak at 0.004.

BEVFusion. For BEVFusion [12] results, we start from the trained PointPillars model weights and fine-tune for other 20 epochs with a cyclic learning rate starting at 10^{-4} with peak at 10^{-3}.

Table 3. Addtional results for ViDAR. Full Range considers a ROI of $[+100m, -100m]$ on the Y axis of the Ego Vehicle and $[+250m, -100m]$ on the X axis, while Short Range restrict the ROI to $51.2m$ on all sides.

History Horizon	Range	CD ↓ $1s$	CD ↓ $3s$
$1s$	Short	15.180	14.915
	Full	58.228	51.720
$3s$	Short	15.204	14.840
	Full	57.284	56.200

SAMFusion. For SAMFusion [15] results, we follow the implementation for the NuScenes dataset reported in the work, removing the Gated-Imaging branch and the Radar branch, essentially keeping only the Camera and LiDAR branches. We initialize the Camera backbone with pre-trained ResNet50 and the LiDAR backbone from scratch. We train the model jointly for 40 epochs with a cosine aneling learning rate, starting from $5 * 10^{-5}$, growing to $5 * 10^{-4}$ in 16 epochs and then dropping to 0.

BEVFormer. For BEVFormer [11] results we rely on the provided codebase implementation. The model is extended to the full ROI of the Long Range Dataset ($-100m : +100m$ on y axis, $-100m : +250m$ on x axis) but the BEV grid size is kept to the original proposed configuration ($200x200$) due to GPU memory limitations. We follow the original implementation and train the model for 20 epochs with a cosine aneling learning rate, with 500 warp-up iterations to $2 * 10^{-4}$.

We additionally test different history horizon and empirically find that using a shorter horizon leads to better results. This is also visible from the drop in perfomance on LiDAR Forecasting on ViDAR model. Tab. 4 shows the mAP and NDS performance on the $1s$ and $3s$ settings with and without ViDAR pre-training.

Table 4. BEVFormer Additional Results.

History Horizon	Pre-Training	mAP ↑	NDS ↑
$1s$	ImageNet	19.40	30.40
	ViDAR	25.32	29.56
$3s$	ImageNet	15.53	25.97
	ViDAR	13.21	23.97

5. Additional Quantitative Results

In the following section we provide additional quantitative results.

5.1. Object Detection

In Tab. 5 we report voxel sizes, memory footprint and inference run time measured on a Nvidia A100 for pytorch native float32 implementations and binned mAP - Short (SR 0:50m), Medium (MR 50:150m) and Long (LR 150:250m) Range. We note that the attention layers in [21] require substantial memory (nuScenes: ∼29Gb, LongRange: >40Gb) during training, limiting voxel size.

Table 5. Extended LongRange Object Detection Results

Method	Mode	Voxel Size [m]	Mem↓ [Gb]	Time↓ [ms]	mAP↑ Full	mAP↑ SR	mAP↑ MR	mAP↑ LR
PointPillars [9]	L	0.195×0.195×12	**1.1**	**51**	39.31	48.2	32.1	31.0
BEVFormer [11] (w/ Pre-train)	C	1.75×1.0×12	2.1	212	24.51	43.8	12.8	0.3
BEVFusion [12]	L+C	0.195×0.195×12	5.7	148	40.10	45.6	39.7	38.0
SAMFusion [15]	L+C	0.195×0.195×12	4.6	402	41.55	44.8	41.5	<u>43.1</u>
FSDv2 [4]	L	0.195×0.195×0.6	3.3	558	38.23	43.7	34.6	27.1
Far3D [6]	C	0.2×0.2×0.2	2.2	157	14.48	33.3	13.4	0.2
SparseFusion [21]	L+C	0.4375×0.25×0.6	3.8	289	22.44	32.9	20.1	14.2
LRS4Fusion (w/o Pre-train)	L+C	0.195×0.195×0.375	<u>1.9</u>	<u>144</u>	<u>49.58</u>	<u>60.9</u>	<u>46.2</u>	40.3
LRS4Fusion	L+C	0.195×0.195×0.375	<u>1.9</u>	<u>144</u>	**52.61**	**63.6**	**48.6**	**43.4**

5.2. Modality Ablation on the LiDAR forecasting task

We ablate the two sensor modalities on the LiDAR forecasting task on the LongRange set on the Near (NFCD 0m to 50m) and Long (LFCD 50m to 250m) Field Chamfer Distance in Tab. 6

Table 6. Modality Ablation on the LiDAR forecasting task

Modality	NFCD ↓	LFCD ↓
Camera	6.163	39.296
LiDAR	3.389	17.338
Fusion	**2.547**	**11.941**

6. Additional Qualitative Results

We present additional qualitative results showcasing latent features, LiDAR forecasting, velocity prediction, and object detection.

6.1. Latent Features

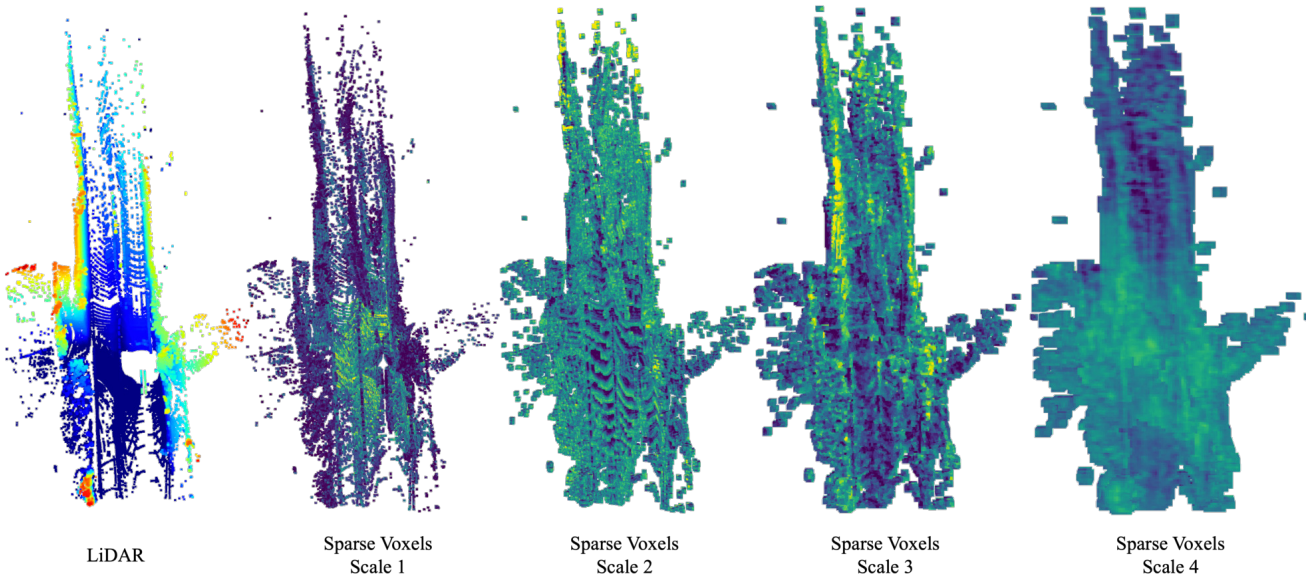

Figure 5. Representation of hidden voxels at different scales extracted by the proposed architecture on the Long Range Dataset. LiDAR Point Cloud for reference. Earlier voxel scales have smaller voxels, capturing finer details with more specialized information, while later scales encompass a larger area, providing a more general representation of the scene.

6.2. LiDAR Forecasting

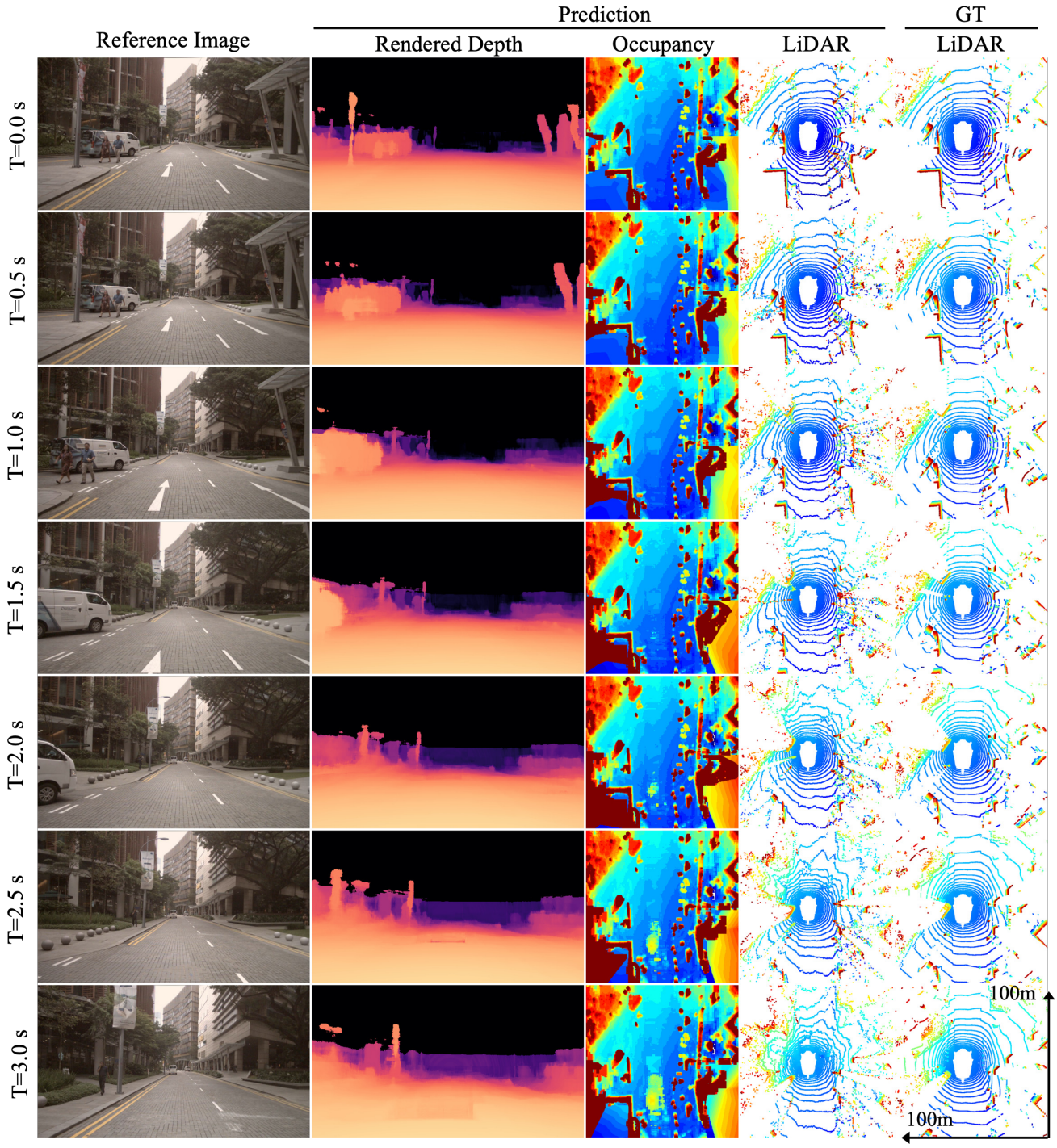

Figure 6. Occupancy and LiDAR reconstruction performances on the NuScenes Dataset. The model is able to accurately reconstruct future LiDAR scans from the predicted 4D occupancy. The Rendered Depth is generated by sampling the occupancy decoder along pixel ray directions and calculating the distance to the first occupied voxel for every ray.

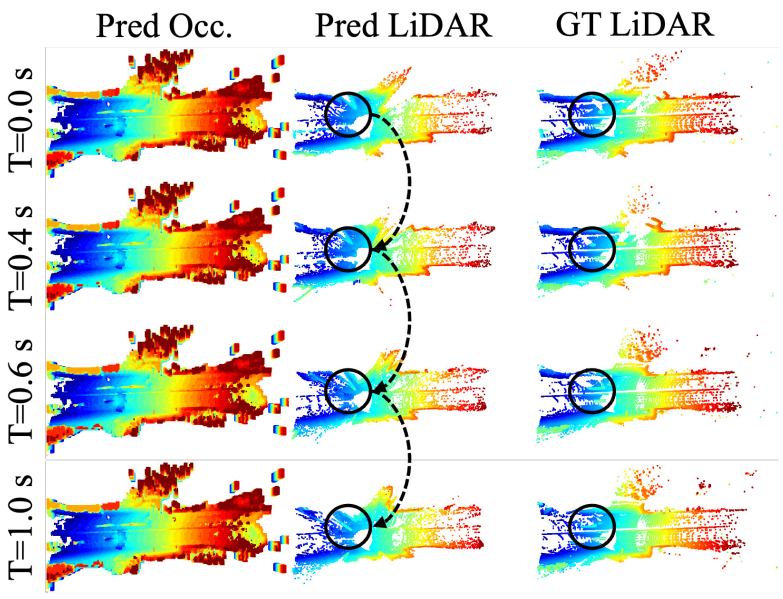

Figure 7. Occupancy and LiDAR reconstruction performances on the Long Range Dataset. The model is able to reconstruct future LiDAR scans from the current frame, correctly predicting vehicle movements in the future.

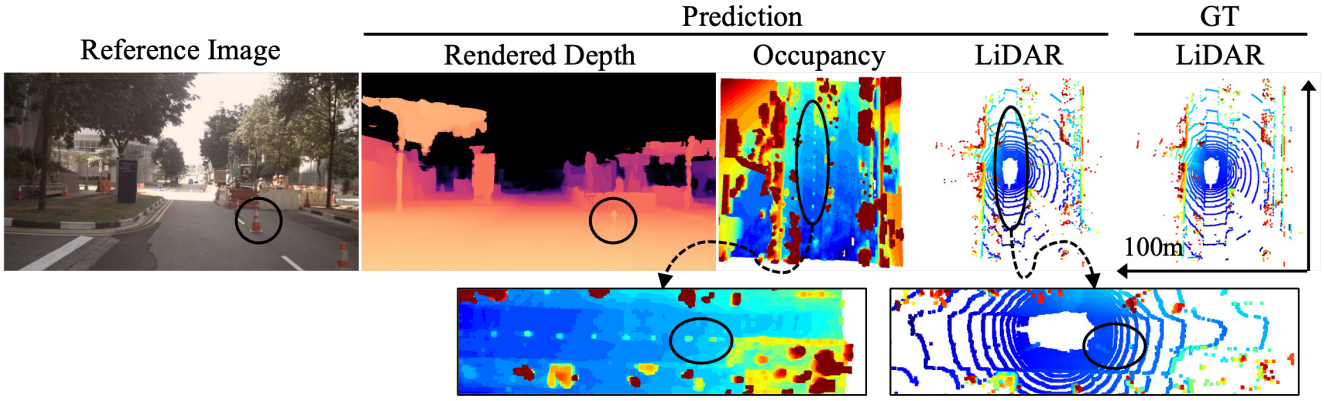

Figure 8. Occupancy and LiDAR reconstruction performances on the NuScenes Dataset. The Proposed model is able to perceive occupancy for small objects in the scene such as traffic cones. The Rendered Depth is generated by sampling the occupancy decoder along each pixel ray directions and calculating the distance to the first occupied voxel for every ray.

6.3. Velocity Prediction

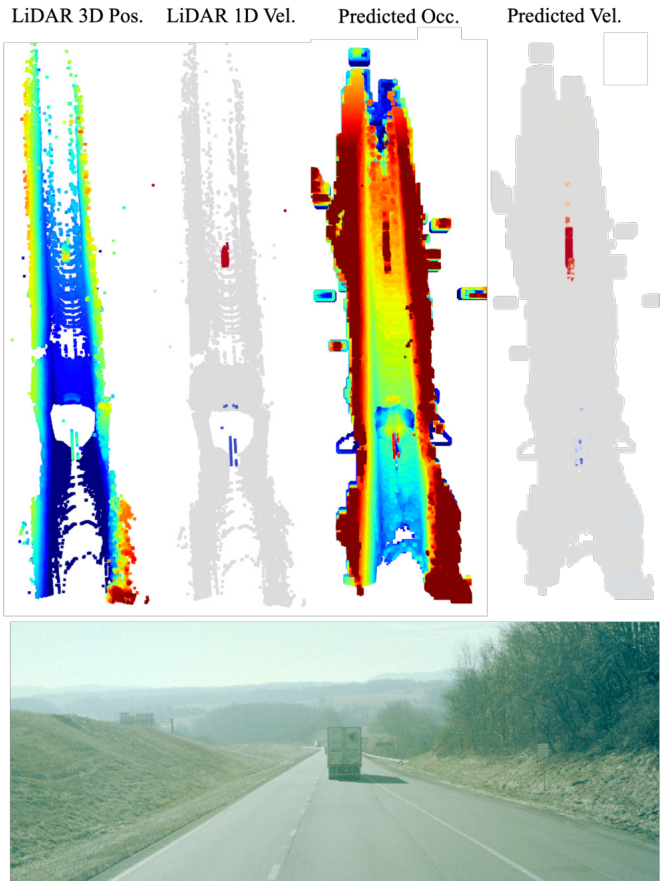

Figure 9. Occupancy and Velocity prediction performances on the Long Range Dataset. The proposed model reconstructs the FMCW velocity of the 4D LiDAR and generalizes it to all points in space beyond the input. From left to right are shown: LiDAR input, LiDAR velocity input, occupancy and predicted velocity.

6.4. Object Detection

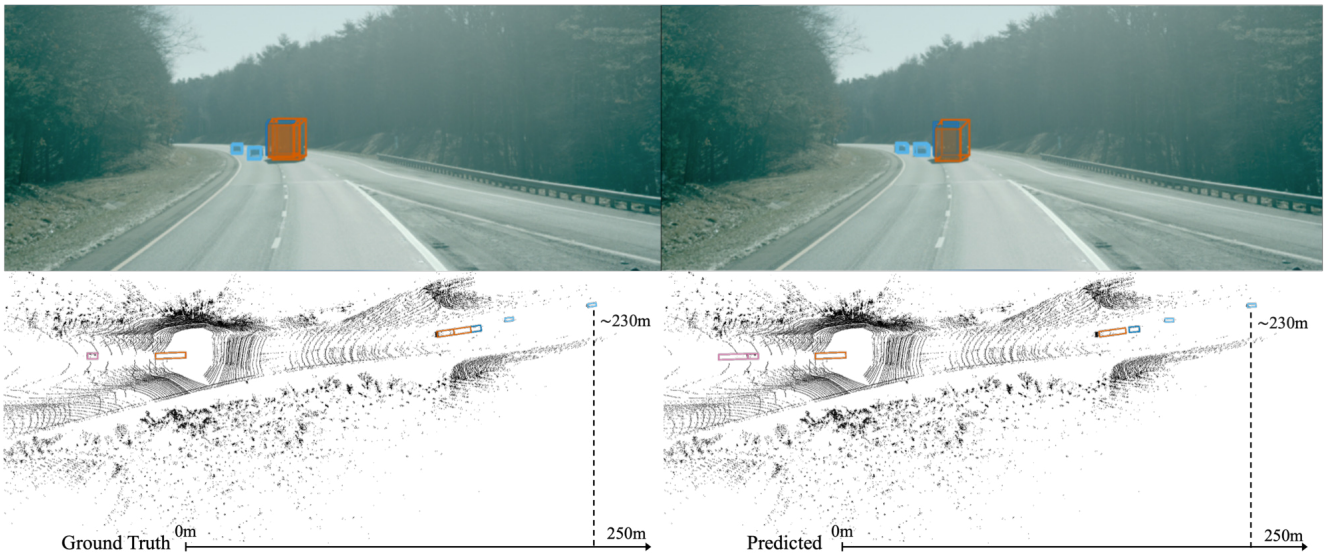

Figure 10. Examples of 3D object detections on the Long Range validation set.

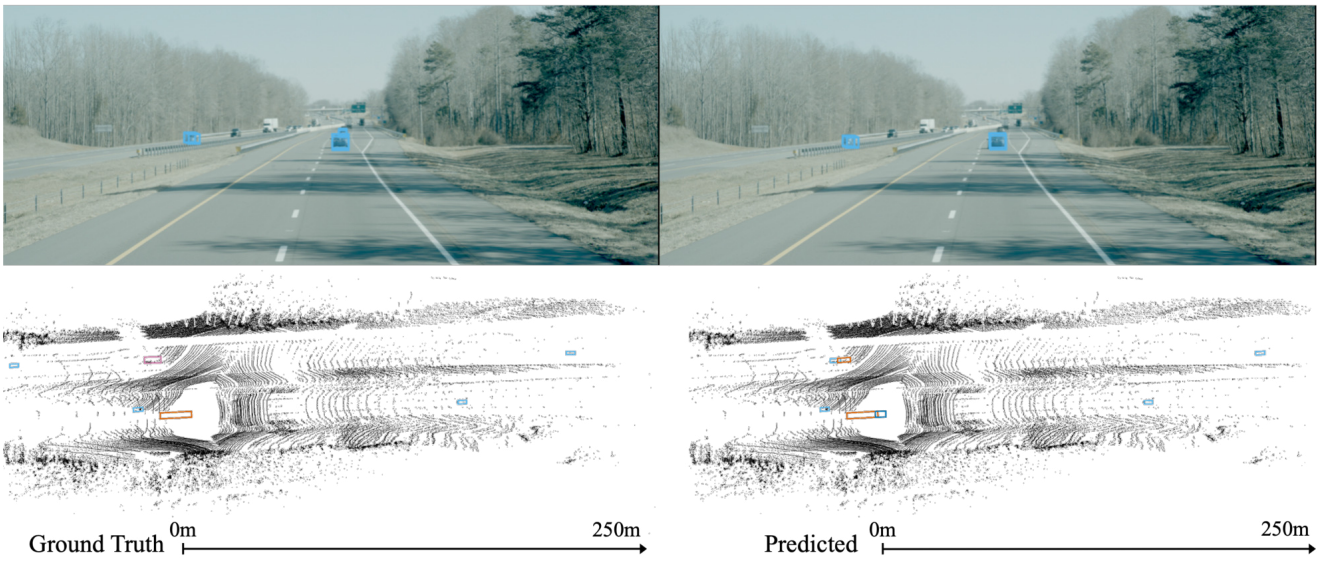

Figure 11. Examples of 3D object detections on the Long Range validation set.

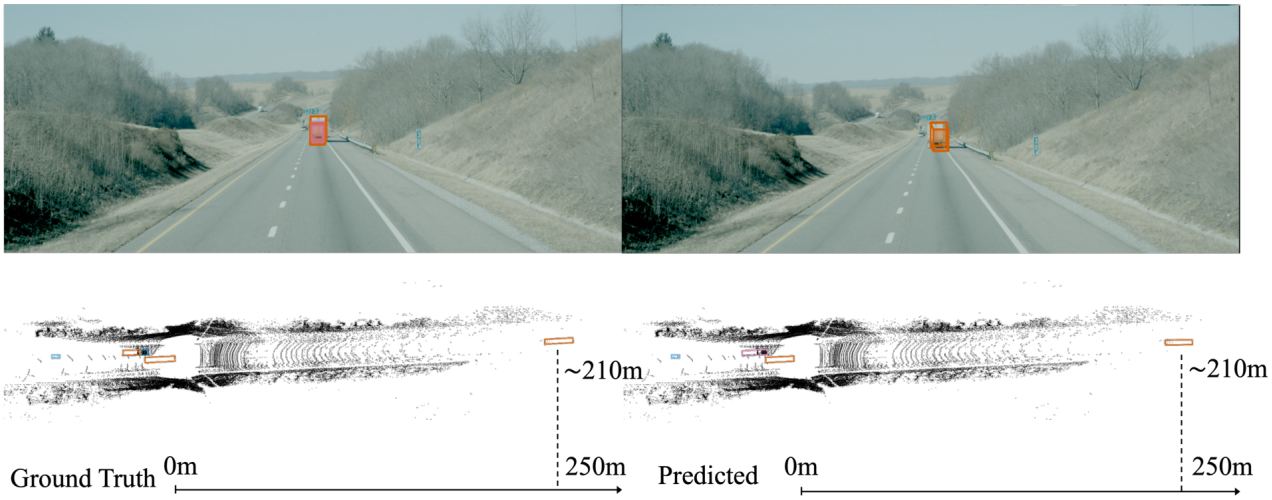

Figure 12. Examples of 3D object detections on the Long Range validation set.

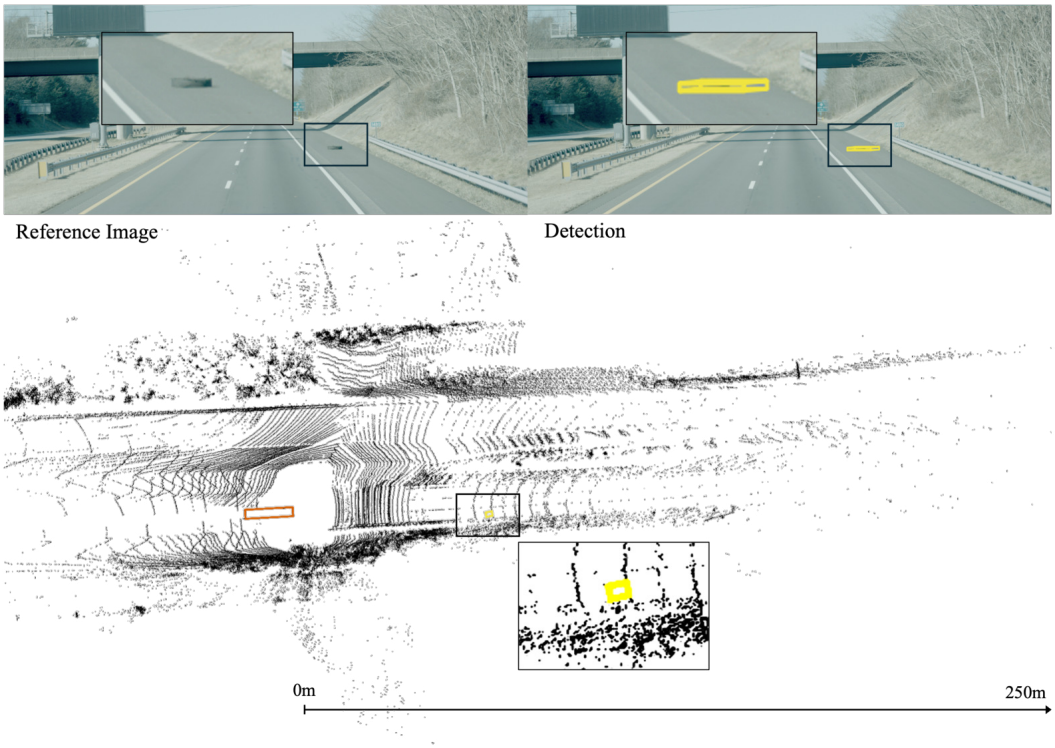

Figure 13. The proposed model is able to detect small lost cargo in the Long Range Dataset. 3D bounding boxes of Road Obstruction with yellow colors.

References

[1] Ben Agro, Quinlan Sykora, Sergio Casas, Thomas Gilles, and Raquel Urtasun. Uno: Unsupervised occupancy fields for perception and forecasting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14487–14496, 2024. 9

[2] 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, 2020. 4, 5

[3] Spconv Contributors. Spconv: Spatially sparse convolution library. https://github.com/traveller59/spconv, 2022. 6

[4] Lue Fan, Feng Wang, Naiyan Wang, and Zhaoxiang Zhang. Fsd v2: Improving fully sparse 3d object detection with virtual voxels, 2023. 12

[5] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012. 4

[6] Xiaohui Jiang, Shuailin Li, Yingfei Liu, Shihao Wang, Fan Jia, Tiancai Wang, Lijin Han, and Xiangyu Zhang. Far3d: Expanding the horizon for surround-view 3d object detection, 2023. 12

[7] Tarasha Khurana, Peiyun Hu, David Held, and Deva Ramanan. Point cloud forecasting as a proxy for 4d occupancy forecasting, 2023. 11

[8] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *2016 Fourth international conference on 3D vision (3DV)*, pages 239–248. IEEE, 2016. 8

[9] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12697–12705, 2019. 11, 12

[10] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints, 2019. 8

[11] Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Yu Qiao, and Jifeng Dai. Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In *European conference on computer vision*, pages 1–18. Springer, 2022. 12

[12] Zhijian Liu, Haotian Tang, Alexander Amini, Xinyu Yang, Huizi Mao, Daniela L Rus, and Song Han. Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2774–2781. IEEE, 2023. 11, 12

[13] B. Mersch, X. Chen, I. Vizzo, L. Nunes, J. Behley, and C. Stachniss. Receding Moving Object Segmentation in 3D LiDAR Data Using Sparse 4D Convolutions. *IEEE Robotics and Automation Letters (RA-L)*, 7(3):7503–7510, 2022. 10

[14] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. 8

[15] Edoardo Palladin, Roland Dietze, Praveen Narayanan, Mario Bijelic, and Felix Heide. Samfusion: Sensor-adaptive multimodal fusion for 3d object detection in adverse weather. In *Proceedings of the IEEE European Conference on Computer Vision (ECCV)*, 2024. 12

[16] Mike Ranzinger, Greg Heinrich, Jan Kautz, and Pavlo Molchanov. Am-radio: Agglomerative vision foundation model – reduce all domains into one, 2024. 8

[17] Tixiao Shan, Brendan Englot, Drew Meyers, Wei Wang, Carlo Ratti, and Rus Daniela. Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5135–5142. IEEE, 2020. 10

[18] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Sheng Zhao, Shuyang Cheng, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset, 2020. 4

[19] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 8

[20] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, Deva Ramanan, Peter Carr, and James Hays. Argoverse 2: Next generation datasets for self-driving perception and forecasting, 2023. 4, 5

[21] Yichen Xie, Chenfeng Xu, Marie-Julie Rakotosaona, Patrick Rim, Federico Tombari, Kurt Keutzer, Masayoshi Tomizuka, and Wei Zhan. Sparsefusion: Fusing multi-modal sparse representations for multi-sensor 3d object detection, 2023. 12

[22] Zetong Yang, Li Chen, Yanan Sun, and Hongyang Li. Visual point cloud forecasting enables scalable autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14673–14684, 2024. 11

[23] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3d object detection and tracking, 2021. 9

[24] Wei Yin, Chi Zhang, Hao Chen, Zhipeng Cai, Gang Yu, Kaixuan Wang, Xiaozhi Chen, and Chunhua Shen. Metric3d: Towards zero-shot metric 3d prediction from a single image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9043–9053, 2023. 8

[25] Youmin Zhang, Xianda Guo, Matteo Poggi, Zheng Zhu, Guan Huang, and Stefano Mattoccia. Completionformer: Depth completion with convolutions and vision transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 18527–18536, 2023. 11

[26] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points, 2019. 8

[27] Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision mamba: Efficient visual representation learning with bidirectional state space model. *arXiv preprint arXiv:2401.09417*, 2024. 8

[28] Yiming Zuo and Jia Deng. Ogni-dc: Robust depth completion with optimization-guided neural iterations. *Proceedings of the IEEE European Conference on Computer Vision (ECCV)*, 2024. 8, 11