UniLiPs: Unified LiDAR Pseudo-Labeling with Geometry-Grounded Dynamic Scene Decomposition (Supplementary Information)

Filippo Ghilotti[1] Samuel Brucker[1] Nahku Saidy[1]
Matteo Matteucci[2] Mario Bijelic[1,3] Felix Heide[1,3]

[1]TORC Robotics [2]Politecnico of Milan [3]Princeton University

https://light.princeton.edu/unilips

This supplemental document provides additional information to support the findings in the main manuscript. Specifically, in Section 1, we present our experimental long-range highway dataset. In Section 2, we provide additional information on the implementation of our auto-labeling pipeline and specifically our bounding boxes optimization, giving more context to the tracking problem and the spline optimization method, showing how we solve the quadratic optimization problem and how this affects the bounding box behavior. In Section 3, we give details on how we trained off-the-shelf models for each task as well as all the baselines we compared against. In Section 4, we extensively evaluate our semantic pseudo labels by comparing them to the ground truth of the SemanticKITTI [2] dataset. In Section 5 we complement the ablation experiments of the main manuscript with additional ones, focusing on depth ablations, moving objects segmentation evaluation with FMCW and parameters settings. Finally in Section 6 we show additional intermediate results and visual examples of our pseudo labels.

Contents

1. Long-Range Experimental Highway Dataset **3**

2. Pipeline Implementation Details **3**
2.1. Pose Estimation (f_{pose}) . . . 4
2.2. Moving Object Segmentation (f_{mos}) . . . 4
2.3. Mapping Module (f_{map}) . . . 4
2.4. Sparse LiDAR 360 Label Recovery . . . 5
2.5. Kalman-Filter Based Tracking . . . 5
2.6. Spline Quadratic Optimization . . . 6

3. Baselines Implementation Details **8**
3.1. Depth Estimation . . . 8
3.2. Semantic Segmentation . . . 9
3.3. Object Detection . . . 9

4. Additional Evaluation and Details on Semantic Pseudo Labeling **9**
4.1. Semantic Pseudo Labels Details . . . 9
4.2. Pseudo Labels Additional Evaluation . . . 9

5. Additional Ablations **10**

6. Additional Results **13**

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

We capture and process a comprehensive long-range dataset specifically for long-haul trucking. Existing public datasets, such as KITTI [7], NuScenes [3], and the Waymo Open Dataset [19] predominantly rely on short-range LiDAR sensors with maximum ranges of approximately 70 to 80 meters. While these datasets have been instrumental in advancing perception algorithms, the limited range of existing LiDAR sensors can constrain a vehicle's ability to detect and respond to distant objects, especially at higher speeds. Our long-range LiDAR dataset provide valuable data for developing and testing algorithms capable of identifying obstacles, traffic signs, and other vehicles from greater distances, as well as allowing deeper depth cues for depth prediction networks. This is crucial for improving the safety and efficiency of autonomous driving systems, enabling better decision-making and longer reaction times in dynamic and complex driving environments. We acquire a long-range dataset from diverse locations in Texas, New Mexico, and Virginia, mainly focusing on highway scenarios to allow long-range perception distances. All sensors are mounted on a semi-truck within a sensor module positioned on top of the driver cabin. The cameras used in our dataset are OnSemi AR0820 models, featuring $1/2$-inch CMOSsensors that capture raw data in RCCB format. Our setup includes synchronized AR0820 cameras recording a near 360 view at $5Hz$ with a resolution of 3848×2168 pixels. For the LiDAR sensors we rely on 4D LiDARs and specifically the AEVA AERIS II, capable of directly measuring radial velocity for each LiDAR point, thanks to Frequency-Modulated Continuous-Wave (FMCW). The velocity of a target is measured by capturing the phase shift between the transmitted and received laser signals. The transmitted signal is a continuous wave with a linearly increasing frequency, known as a chirp. Upon reflection from a moving object, the received signal experiences a time delay τ and a Doppler-induced phase shift $\Delta\phi$ from which a radial velocity can be computed as follow:

$$s_{\text{rx}}(t) = \cos\left(2\pi\left(f_0(t-\tau) + \frac{\kappa(t-\tau)^2}{2}\right) + \Delta\phi\right), \quad v = \frac{\Delta\phi\lambda}{4\pi}, \quad v_r = v\cos\theta \tag{1}$$

where λ is the wavelength of the laser signal and θ is the incident angle between the object's velocity vector and the line-of-sight. They captures data at $10Hz$, with a range of up to $400m$ and are as well displayed to cover $360°$ around the semi-truck. The camera-LiDAR system is also synchronized for cohesive data capture. Manual calibration is performed between recordings to maintain system accuracy. Data collection covers various natural lighting conditions, as presented in Figure 1. The dataset comprises a total of $60,000$ unlabeled frames and $36,000$ manually annotated frames tailored for object detection across seven categories: *Bike*, *Passenger-Car*, *Person*, *RoadObstruction*, *SemiTruck-Cab*, *SemiTruck-Trailer*, *Vehicle*. Due to the nature of the driving scenarios in highway environment and the capabilities of our powerful sensor setup, the detection distribution grants high density in further region too, as shown in figure 2. To ensure accurate annotations, the dataset leverages both camera and LiDAR data in a complementary manner.

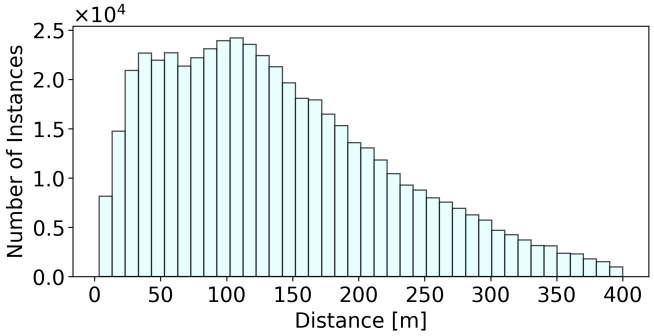


Figure 2. **Instances Distribution** of bounding boxes and detections across ranges

2. Pipeline Implementation Details

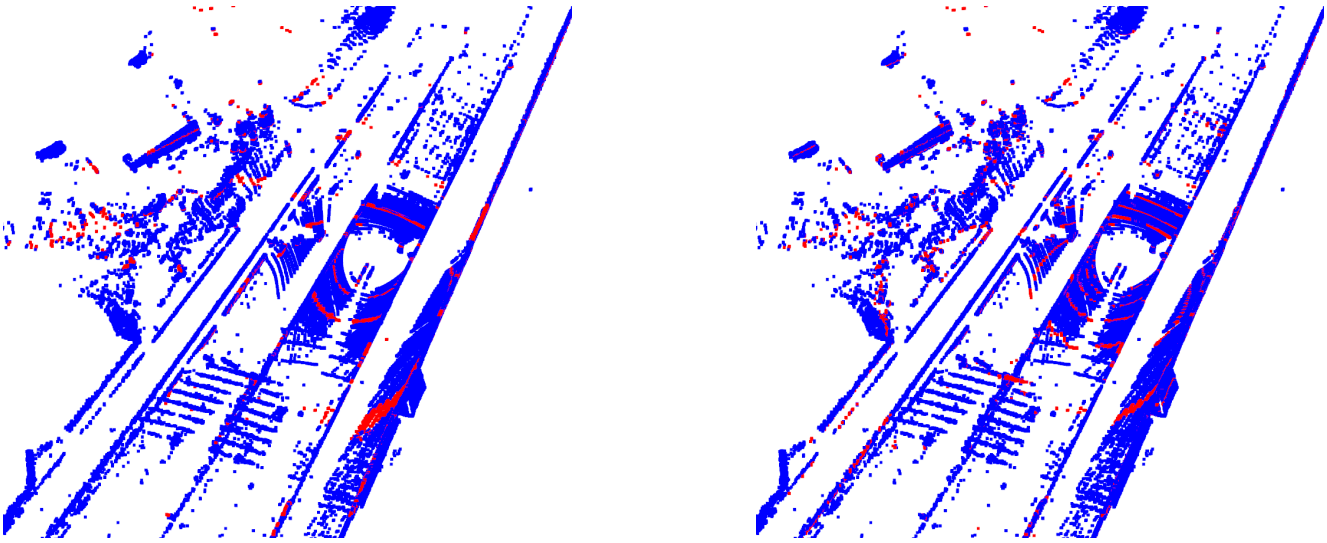
The proposed framework encompasses several core functions, each addressing a specific aspect of the data processing pipeline essential for robust performance in autonomous driving scenarios. These functions include pose estimation (f_{pose}), moving object segmentation (f_{mos}), mapping (f_{map}) and the spline optimizer We present some of the choices made for each function, detailing why a specific method was chosen in this context.

2.1. Pose Estimation (f_{pose})

For our dataset specifically, as well as for any other, we required a method capable of handling the complexities introduced by rapid motion and dynamic objects, as the moving objects segmentation function f_{mos} requires an accurate pose estimation, especially in high-speed and highly dynamic environments. After evaluating several algorithms, we chose PIN-SLAM [17] due to its superior robustness and reliability, particularly in high-speed highway driving scenarios: integrating probabilistic inference with SLAM, it showed enhanced performances in dynamic conditions. Its precision and robustness made it the preferred choice for use across almost the entire dataset. For those few scenes in which the method would fail we found MAD-ICP [5], which introduces a Multi-Agent Distributed ICP framework, to improve scalability and accuracy in complex scenarios, to be a valid alternative to complement PIN-SLAM [17] in retrieving pose estimation for every scene.

2.2. Moving Object Segmentation (f_{mos})

The implementation of f_{mos} follows the methodology outlined in [16], which offers flexibility in adjusting the confidence threshold for detecting dynamic entities. The approach in [16] utilizes a confidence-based thresholding mechanism that allows the system to dynamically adjust its sensitivity to moving objects based on environmental conditions and sensor noise levels. This adaptability ensures that only reliably detected moving objects are segmented, thereby minimizing false positives and allowing the mapping module to focus more on static features of the environment.



(a) 64 Channels (b) 128 Channels

Figure 3. **Visualization of Different Channels Settings.** For the 64 channels setup we plot the channels 10, 20, 30, 40, 50, 60 showing their high spatial separation and thickness, while for the 128 channels setup we add channels 70, 80, 90, 100, 110, 120 for comparing the mapped space by each channel.

2.3. Mapping Module (f_{map})

The mapping component of the framework is implemented using the LIO-SAM algorithm [18], which provides a tightly-coupled LiDAR-Inertial Odometry and Mapping solution. This module can be modified to operate on the specific sensor setup employed and is deployed without GNSS measurements. Parameters are chosen in such a way that algorithm prioritizes mapping accuracy over real-time processing in order to enhance precision and obtain detailed mapping. Since the algorithm is usually deployed on rotating LiDARs, it requires the channel information, which is not available in the case of our solid state AEVA measurement. To simulate the behavior of a rotating LiDAR using a solid-state LiDAR, we can employ a method that assigns each point in the point cloud to discrete virtual channels based on their elevation angles, thereby emulating the multi-beam 360 scanning pattern of a mechanically rotating sensor without physically rotating the points. Let the vertical field of view of the LiDAR be denoted by Θ, which is discretized into N virtual channels with elevation angles θ_i for $i = 1, 2, \ldots, N$. These angles are uniformly spaced within the range

$$\theta_i = -\frac{\Theta}{2} + \left(i - \frac{1}{2}\right)\frac{\Theta}{N}, \quad i = 1, 2, \ldots, N. \tag{2}$$

For each point with Cartesian coordinates (x, y, z) in the point cloud, we compute its elevation angle ϕ relative to the horizontal plane as

$$\phi = \arctan\left(\frac{z}{\sqrt{x^2 + y^2}}\right). \tag{3}$$

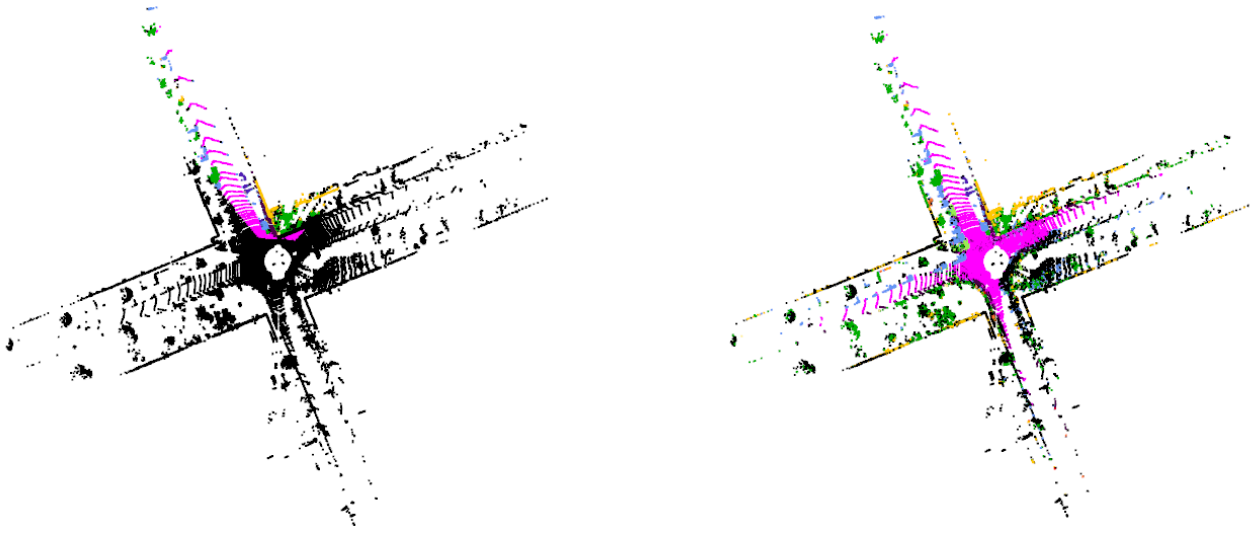
The point is then assigned to the channel j that minimizes the absolute difference between its elevation angle and the predefined channel angles

$$j = \arg\min_k |\theta_k - \phi|. \tag{4}$$

We find that simulating 128 horizontal scans allows to not assign too many points to the same channel, which would result in a worse feature matching from the mapping module [18]: in Figure 3 we show this by comparing with a 64 layers simulation, where channels are thickened and far from each other, hindering the mapping process.

2.4. Sparse LiDAR 360 Label Recovery

Figure 4 illustrates an example using the SemanticKITTI [2] dataset, presenting results in a before-and-after fashion. Initially, we assign semantic labels to the LiDAR data using only the front left camera. Since the two stereo images overlap by approximately 90%, incorporating the right camera offers significant benefits primarily during the initial few time steps. To generate 360-degree pseudo labels, we align each LiDAR scan to the accumulated label map using the pose estimate $\mathcal{T}'$ and assign labels to each point by comparing it to surrounding map points within a spherical neighborhood of 50 centimeters. Due to the limitation on label propagation, some regions in the map — and consequently some points in the LiDAR scans — remain unlabeled. For this reason, when training our model and baselines, we exclude the first 10 pseudo-labeled frames of each sequence because, unless the car revisits the starting point later in the sequence (as in a loop), these frames cannot be labeled in all directions.



(a) Labels Initialization (b) Labels Re-Weighting

Figure 4. **Kitti 360 Label Retrieval.** Due to camera view-only labeling, initially, LiDAR labels (left) are only present in the front: by re-comparing the lidar scan with the labeled map and re-weighting in a neighborhood of the single scan point it is possible to retrieve 360 degrees labeling of the pointcloud (right).

2.5. Kalman-Filter Based Tracking

Operating recursively on streams of noisy input data to produce statistically optimal estimates of the underlying system state, the Kalman filter provides an efficient method to estimate the state of a process in a way that minimizes the mean squared error. We model the system based on a constant velocity model, assuming object's velocity remains the same between consecutive time steps, which is a reasonable approximation over short intervals in many real-world scenarios: specifically

LiDAR-based system often running at 10 Hz, resulting in a recording every 100 milliseconds. Each object state is represented by a state vector x that includes its position, size, orientation, and velocities

$$x = \begin{bmatrix} x & y & z & l & w & h & \theta & \dot{x} & \dot{y} & \dot{z} & \dot{\theta} \end{bmatrix} \tag{5}$$

We initialize a high state covariance matrix for velocities, as we do not exploit velocity measurements. Upon receiving new detections, the tracker module tries to match them to existing trackers using the Hungarian algorithm based on the 3D Intersection over Union (IoU). If a match is possible it updates trackers with assigned detections, otherwise new trackers for unmatched detections are created. Following the literature [21, 22], as, specifically in LiDAR, objects may be occluded for few timestamps or the pseudo bounding box may be missing, unmatched trackers are not deleted and the prediction step is done seamlessly: in this way, each timestamp the model tries to recover the lost track even if the Kalman Filter was not updated. Since evaluation algorithms for bounding box detections usually require the knowledge of a confidence score we exploit the uncertainty in the position estimates from the Kalman filter. A lower covariance indicates higher confidence. The score s for each object is then computed as

$$s = \frac{1}{\sqrt{\det(\mathbf{P}_{\text{pos}}) + \epsilon}} \tag{6}$$

where $\mathbf{P}_{\text{pos}}$ is the covariance submatrix for position (x, y, z), and ϵ is a small constant to prevent division by zero.

2.6. Spline Quadratic Optimization

After tracking bounding box we are able to generate trajectories: due to noise however, these do not represent a feasible, phisical trajectory of rigid objects. For this reason we use a phisical spline to optimize the position and the yaw of object and induce realistic and, thus, more accurate trajectories

Yaw Optimization. We represent the yaw as a combination of basis functions, modeling both the sine and cosine components of the yaw angle, ψ. The objective of the yaw optimization is to minimize the following cost function, e_{yaw}, which penalizes the difference between the estimated and measured yaw directions

$$e_{\text{yaw}} = \frac{1}{2} \sum_j \Big(\left(f_c(t_j) - \cos(\psi_j)\right)^2 + \left(f_s(t_j) - \sin(\psi_j)\right)^2 \Big) \tag{7}$$

where $f_c(t)$ and $f_s(t)$ are the model's estimates for the cosine and sine of the yaw angle, represented as linear combinations of basis functions

$$f_c(t) = \sum_k w_{\cos,k} \cdot f_k(t), \quad f_s(t) = \sum_k w_{\sin,k} \cdot f_k(t) \tag{8}$$

and ψ_j is the measured yaw angle at each time t_j. The yaw angle at each time t can be reconstructed using the arctangent

$$\psi(t) = \arctan 2(f_s(t), f_c(t)). \tag{9}$$

To enforce smooth, coherent motion, the second derivatives of $f_c(t)$ and $f_s(t)$ are modeled as first-order B-splines. This problem is then solved using quadratic optimization. To express the problem as a quadratic optimization problem, we construct the matrices Q and b based on the contributions of each time point t_j. Specifically, we have

$$Q = \sum_j Q_j, \quad b = \sum_j b_j, \tag{10}$$

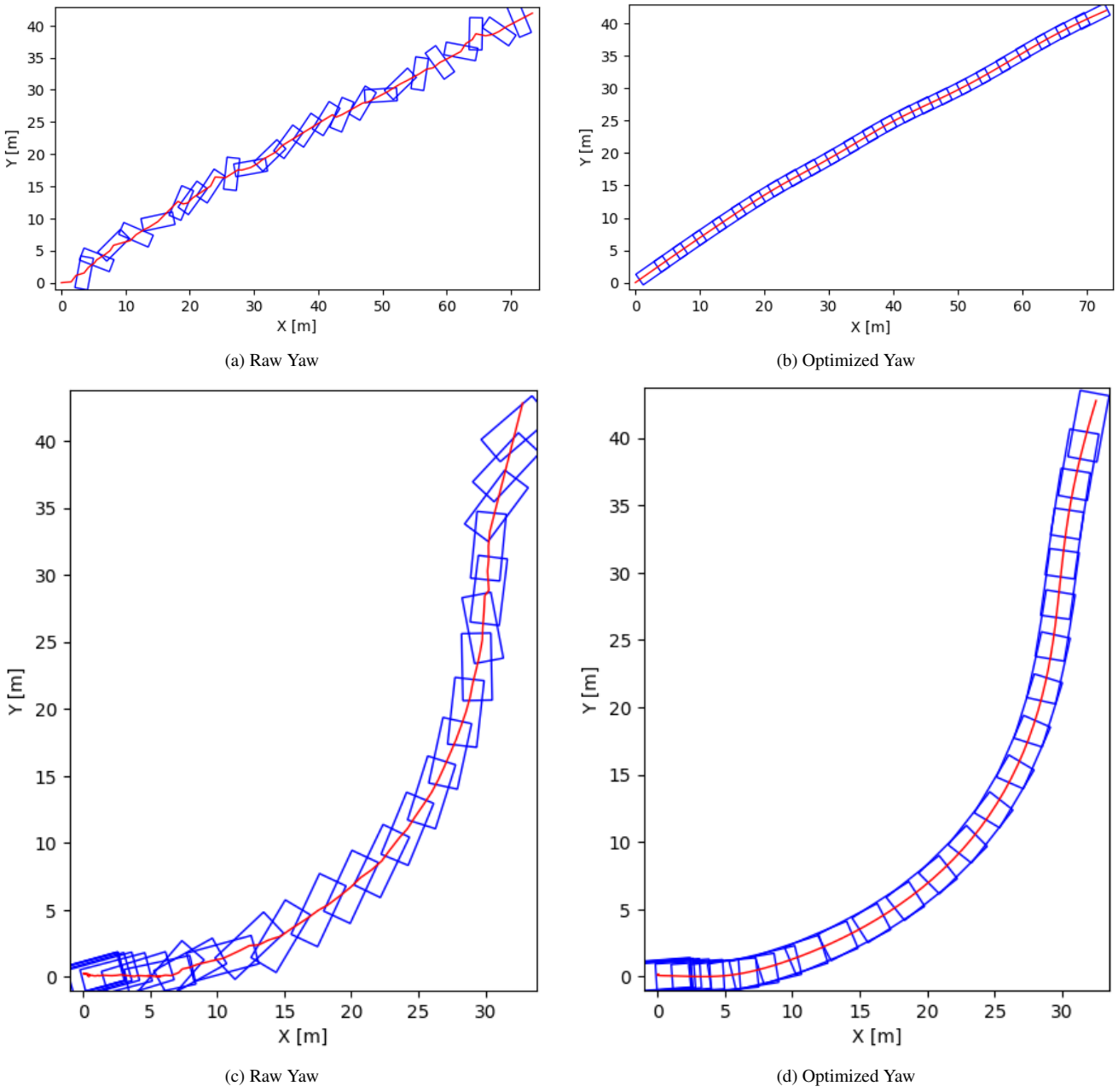
where Q_j and b_j are derived from the linear system formulation for each measurement. This leads to the following linear system representation of the yaw optimization problem

$$Q \cdot w = b, \tag{11}$$

where w contains the weights of the basis functions and Q and b are the system matrices that aggregate the contributions from all the measurements.

Weight Vector. The weight vector w is the concatenation of $w_{\cos}$ and $w_{\sin}$, where

$$w_{\cos} = \left(y_0,\, \dot{y}_0,\, \ddot{y}_{\Delta t_1},\, \ddot{y}_{\Delta t_1 + \Delta t_2},\, \dots\right), \tag{12}$$



(a) Raw Yaw

(b) Optimized Yaw

(c) Raw Yaw

(d) Optimized Yaw

Figure 5. **Effect of Spline Optimization.** We illustrate how the spline optimization affects the yaw and position of the bounding boxes, from their raw format (a,c) to the optimized one (b,d). We show results for two scenes (a,b) and (c,d). The trajectories get effectively smoothed and bounding boxes get beeter aligned to the trajectory throughout all subsequent timestamps.

$$w_{\sin} = (x_0,\, \dot{x}_0,\, \ddot{x}_{\Delta t_1},\, \ddot{x}_{\Delta t_1 + \Delta t_2},\, \ldots)\,. \tag{13}$$

The weight vectors $w_{\sin}$ and $w_{\cos}$ represent the parameters for the sine and cosine components of the yaw, respectively, at each time step. These parameters include the initial component values, the rate of change, and the second-order rate of change for each time step.

Adding Regularization. To ensure a smooth and stable solution, we add regularization terms to the cost function to smooth

the yaw estimate. We apply a regularization based on the second derivative of the basis function. This can be expressed as

$$e_{\text{reg},0} = \frac{1}{2}\sum_k w_k^2, \qquad e_{\text{reg},1} = \frac{1}{2}\sum_k (w_{k+1} - w_k)^2. \tag{14}$$

The total regularized cost function becomes

$$e_{\text{total}} = e_{\text{yaw\_total}} + \alpha\, e_{\text{reg},0} + \beta\, e_{\text{reg},1}, \tag{15}$$

where $\alpha = 0.01$ and $\beta = 0.02$, were determined empirically to control the influence of the regularization terms on the overall optimization.

Solving the Linear System. The regularized system can be represented as

$$(Q + Q_{\text{reg}}) \cdot w = b, \tag{16}$$

where Q_{reg} includes the regularization terms. Solving this system for w yields the optimal parameters, resulting in a smooth yaw estimate.

Position Optimization. To optimize objects position, we apply a similar approach. Instead of estimating the sine and cosine components of the yaw, the method can directly estimate the x and y positions. The position error function is defined as

$$e_{\text{position}} = \frac{1}{2}\sum_j \left((f_x(t_j) - x_j)^2 + (f_y(t_j) - y_j)^2 \right), \tag{17}$$

where x_j and y_j represent the measured positions at time t_j.

Visual Effect Of Trajectory Optimization. Figure 5 presents a comparison between two non-optimized and optimized results.

3. Baselines Implementation Details

In this section we provide more information on the choices of the off the shelf model trained to evaluate the quality of our pseudo labels as well as the baseline we compared against

3.1. Depth Estimation

NMRF[8]. For depth estimation we selected NMRF as it demonstrated state-of-the-art performance on the KITTI benchmark and strong generalization capabilities. Its implementation is very simple and easy to reproduce and it comes with a public checkpoint, pre-trained on synthetic data. For KITTI Dataset we additionally used the available checkpoint fine-tuned on sparse LiDAR scans, while for our long range dataset we followed the same configuration reported in their paper, fine-tuning on our sparse LiDAR for 30k steps randomly cropping images to 304 × 1152 and using a maximum learning rate of 0.0002.

We then fine-tuned NMRF with different dense pseudo-labels, keeping the hyper-parameters fixed

$$l_{rate} = 0.002, Crop\_Size = [304, 1152], Loss\_Type = SMOOTH\_L1 \tag{18}$$

and changing only the baseline and focal length parameter based on the dataset used. As reported in the main manuscript, we rely on reverse Huber [11, 26] (berHu) loss: after defining $\delta_i = |\text{pred}_i - \text{target}_i|$ and $c = 0.2 \times \max_i |\text{pred}_i - \text{target}_i|$

$$L_i = \begin{cases} \delta_i, & \text{if } \delta_i < c \\ \frac{\delta_i^2 + c^2}{2c}, & \text{if } \delta_i \geq c \end{cases}, \quad \mathcal{L} = \frac{1}{N}\sum_{i=1}^{N} L_i \tag{19}$$

Lio-Sam [18]. The first comparison is performed finetuning NMRF on accumulated LiDAR data: we used Lio-Sam for simplicity, as already implemented in our f_{map}, as a viable alternative to generate a dense LiDAR ground-truth. However, we used the full scan set S without removing moving objects from MOS_{init} and didn´t apply our *Adaptive Spherical Occlusion Culling*. This let us understand the importance of map optimization during accumulation, as well as of floaters and artifacts removal (showed qualitatively in Figure 8) in post-processing for accurate depth.

CREStereo [13]. The second comparison is carried out finetuning NMRF on predictions from a stereo method, which showed powerful generalization capabilities. We used the available pre-trained model and inferred on our training splits (from the 2 datasets), to generate the pseudo ground-truth, which we used instead of the projected LiDAR.

Depth Anything V2 [24]. Lastly we compared finetuning NMRF with the prediction generated by a foundation model. We use the available *Depth-Anything-V2-Large*, finetuned on outdoor dataset to predict metric depth estimation and, as for the previous comparison, inferred on our training splits (from the 2 datasets), to generate the pseudo ground-truth, which we used instead of the projected LiDAR.

3.2. Semantic Segmentation

PVKD [9]. We select PVKD [9] because its approach of distilling knowledge first through voxels and then through points aligns well with our sparser observations. We train the model, for all the experiment, for maximum 20 epochs with a learning rate of 0.002 and report the best score on evaluation sequence 08 of SemanticKITTI.

LeAP [6]. In order to compare with LeAP we rely on the results published on their main paper, as no code is yet available. To be fair we use the same evaluation settings detailed in their manuscript, considering only labeled point in the evaluation and grouping semantic classes in 2 reduced sets, defined in TABLE 7 of the benchmark paper of KITTI 360 [14].

Semantic SAM [4]. For semantic SAM we use the public pseudo-labeling engine and project the semantics from image to LiDAR, then propagate the labels and recover 360 degrees segmentation in the same way we do for our engine.

Pre-Trained [25]. is a Cylinder3D model, trained on NuScenes and fine-tuned on 2000 samples of the SemanticKITTI GT-data, used to infer pseudo-labels, which are subsequently used to train the PVKD model, similarly to what is done by LeAP to simulate a situation in which a new dataset is acquired and no labels is available.

Lasermix Vx[10] is trained with vanilla settings by replacing the Cilynder3D backbone with a PVKD backbone.

3.3. Object Detection

PointPillars [12]. We train the public implementation of PointPillars, on the presented mixes of ground-truth and pseudo labels, in all cases: for 20 epochs with a cyclic learning rate starting at 0.00025 with peak at 0.004. We limit the range to a maximum grid of $[-70m, +70m]$ due to memory limitations and infer on the same range.

LISO [1]. We train and run LISO inference with available fully unsupervised settings, based on flow estimation on $250m$ range.

ICP-Flow [15]. We run ICP-Flow after tuning the HDBSCAN module. The model inherently requires a pose ground-truth or it estimates it using lidar odometry, specifically KISS-ICP [20]. Since, as previously reported, this method fails consistently on our dataset, we instead use our better pose estimate coming from LiDAR Inertial Odometry ([18]).

4. Additional Evaluation and Details on Semantic Pseudo Labeling

4.1. Semantic Pseudo Labels Details

Table 1 presents the per-class results of our pseudo labels (extending Table 2 of the main paper), evaluated following the LeAP methodology [6]. Additionally, we evaluate the pseudo labels generated from Cylinder3D [25] (Pre-Trained) trained on NuScenes [3] and fine-tuned on 2000 frames from SemanticKITTI [2], that we used to simulate automatic labeling of unseen data. Finally in Table 2 we extend Table 3 of the main manuscript presenting per clas IoU %.

4.2. Pseudo Labels Additional Evaluation

To quantitatively assess the quality of our pseudo semantic labels on the SemanticKITTI dataset [2], in Table 3 we extend the reduced classes evaluation to all the scenes in the dataset (excluding the already reported scene 08). Eventually, in Table 4 we evaluate the per-sequence performance using overall accuracy, mean Intersection over Union (mIoU), and a weighted mIoU (where weights reflect the per-label point count in each scene), considering all the points in each scan. Additionally, we report the percentage of LiDAR points that receive a semantic label under our method. To highlight the benefits of label propagation and re-weighting — especially for recovering labels in regions without pixel correspondences and improving mIoU — we compare against the pseudo-labels produced by our engine and emphasize the resulting relative gains.

Distillation Benefits. In table 5 we used our pseudo labels to train the self-supervised model Lasermix [10] and a knowledge distillation method like ReDal [23] and found that also these methods can benefit from a small amount of our refined pseudo labels: we report improvements over the base model in terms of mIoU and the top 2 classes, by adding 20% of our pseudo labels to the 10% ground truth used.

Method	mIoU %	cat-mIoU %	Automotive IoU %											Categories IoU %					
			car	bicycle	motorcycle	oth.-veh.	person	road	sidewalk	oth.ground	manmade	vegetation	terrain	flat	construction	object	nature	human	vehicle
Pre-Trained [25]	40.5	68.3	84.7	0.0	1.6	1.0	9.6	79.6	60.0	0.0	**80.6**	78.4	50.5	88.9	81.6	57.9	86.7	9.4	85.6
LeAP (Points) [6]	46.8	68.6	77.2	25.5	15.1	30.3	31.9	87.1	46.1	0.0	64.3	64.7	**72.3**	-	-	-	-	-	-
Our f_{seg}	59.4	69.6	80.7	38.3	40.9	66.5	**57.6**	90.4	60.0	0.0	72.3	82.4	63.9	92.4	73.1	41.8	83.2	46.9	80.0
Our Propagated	**64.9**	**76.2**	**93.2**	**40.8**	**67.6**	**78.9**	50.7	**92.1**	**61.7**	0.0	77.5	**88.6**	62.8	93.7	78.3	58.7	88.1	45.0	93.3
LeAP (Voxel + 3D-CN (2))	58.1	81.6	92.5	24.5	26.8	27.3	71.7	93.9	73.7	0.0	69.4	82.9	76.2	-	-	-	-	-	-
Our Voxel Prop.	68.3	86.6	95.1	58.3	60.8	73.1	73.6	92.1	64.8	0.0	80.8	89.8	63.4	96.1	84.3	72.3	93.8	76.6	96.3

Table 1. **LeAP Class Comparison.** We show in details the per-class mIoU of our f_{seg} and our propagated pseudo labels compared to LeAP [6] ones: following their methodology we evaluate only on labeled points for fair comparison and demonstrate the better quality of our pseudo labels. Best results are in **bold**, second best underlined

	Class IoU %																		
Method (GT - Pseudo)	Car	Bicycle	Motorcycle	Truck	Bus	Person	Bicyclist	Motorcyclist	Road	Parking	Sidewalk	Other-ground	Building	Fence	Vegetation	Trunk	Terrain	Pole	Traffic-sign
Oracle (100-0)	94.73	45.34	60.65	50.62	43.75	73.34	83.83	6.40	94.04	44.62	81.13	4.44	89.23	55.23	87.41	68.72	72.72	64.93	51.75
Limited GT (10-0)	91.73	14.75	12.48	15.21	16.36	37.18	49.95	0.00	78.39	26.73	70.15	0.06	74.14	40.90	76.72	50.03	70.84	59.32	39.94
Semantic-SAM (10-90)	80.70	20.88	39.37	10.23	0.23	41.04	17.41	0.00	60.27	0.00	55.57	0.00	78.38	19.18	77.41	18.50	34.86	49.44	37.06
Pre-Trained (10-90)	92.49	15.68	19.67	15.65	11.19	30.79	32.70	0.01	88.87	16.54	72.48	0.11	85.98	49.95	**86.63**	62.81	**72.64**	61.56	45.60
Lasermix (Vx) (10-90)	94.97	43.98	**59.51**	64.72	39.47	65.92	75.64	0.19	92.17	42.30	77.24	**2.65**	89.26	50.09	85.36	62.44	72.43	58.51	**51.29**
Proposed (10-90)	**95.12**	**44.70**	53.03	53.58	**53.78**	67.39	**82.06**	**0.51**	**92.22**	**47.66**	77.39	0.18	**90.18**	**57.29**	86.52	**64.98**	71.38	**65.49**	48.50

Table 2. **Evaluation of Semantic Segmentation for Class-IoU.** We report here IoU percentage results per class on the validation set, training PVKD [9] with different percentages of our pseudo labels. For *10-90* model, we achieve near-Oracle quality, with a small average difference of -0.30%, when considering only mapped classes. In general our semantic labels can enhance the training of the *10-0* model on most of classes, matching the trend of the *Full Pseudo* results. Excluding the oracle, best results are in **bold**, second best are underlined

5. Additional Ablations

Since the capability of our LiDAR of measuring radial velocity is not available for every dataset we deliberately do not rely on this signal in our method, as generalization is a key feature of our processing. However, we can use this measurement to evaluate our moving object segmentation as a binary classification of a point nature of being *moving* or *nonmoving*. To generate a ground truth, velocity measurements are segregated into positive and negative components to account for motion in different directions. For each subset of velocities, the mean (μ^+),(μ^-) and standard deviation (σ^+),(σ^-) are computed. A point is classified as *moving* if its velocity exceeds a dynamically determined threshold, specifically defined as $\mu^+ + 2.0\sigma^+$ for positive velocities and $|\mu^-| + 2.0\sigma^-$ for negative velocities. This statistical approach ensures that only points exhibiting significant motion, relative to the overall velocity distribution, are considered moving, thereby reducing the influence of random noise and minor fluctuations. To further improve the robustness of the labeling process and mitigate the impact of isolated outliers, a neighborhood-based filtering mechanism is employed. For each point initially labeled as *moving*, the spatial vicinity is examined to verify the consistency of its motion classification. Utilizing spatial indexing structures such as KD-tree, a *moving* point retains its label only if it is surrounded by a minimum number of other *moving* points within this neighborhood. If a *moving* point lacks sufficient neighboring *moving* points, it is disregarded as an outlier. This spatial consistency check ensures that motion labels are not only statistically significant but also supported by the local spatial

Sequence	mIoU %	cat-mIoU %	Automotive IoU %											Categories IoU %					
			car	bicycle	motorcycle	oth.-veh.	person	road	sidewalk	oth.-ground	manmade	vegetation	terrain	flat	construction	object	nature	human	vehicle
00	56.48	69.20	82.26	26.62	44.60	39.81	37.89	92.80	73.55	0.00	81.84	80.31	61.64	95.02	81.84	41.40	81.61	33.08	82.28
01	27.84	50.81	72.90	0.00	17.20	0.00	0.00	92.41	0.00	0.00	31.32	50.47	41.98	91.62	26.21	36.01	79.49	0.00	71.52
02	52.59	59.39	82.39	13.64	70.78	51.45	49.27	87.79	62.32	0.00	50.70	81.91	28.25	93.79	50.47	32.26	81.12	18.74	79.97
03	43.82	62.38	82.28	29.92	0.00	58.99	0.00	89.82	67.87	0.00	58.86	57.14	37.15	95.29	58.36	50.11	89.62	0.00	80.90
04	40.59	63.26	87.60	0.00	N/A	0.00	0.00	90.58	16.11	0.00	68.12	84.11	59.40	95.74	67.42	38.53	88.72	0.00	89.14
05	55.45	63.82	79.40	45.40	2.01	78.44	50.90	91.68	72.83	0.00	72.18	74.79	42.27	95.12	71.87	28.00	73.56	34.87	79.52
06	55.94	71.13	76.71	38.91	30.96	63.53	34.90	87.84	58.67	0.00	73.16	65.44	85.26	94.26	79.18	48.73	87.74	33.14	83.74
07	70.77	73.38	83.32	38.83	61.57	77.65	60.30	93.28	71.62	N/A	81.14	77.98	61.95	94.58	81.11	42.97	83.90	54.70	83.00
09	54.32	70.81	77.34	7.97	53.73	56.37	50.73	89.84	65.36	0.00	68.29	76.15	51.70	96.28	68.85	39.82	86.70	51.25	81.94
10	51.96	67.62	84.58	0.00	28.61	86.48	38.81	85.10	43.82	0.00	69.48	78.48	56.25	94.97	68.96	34.89	82.84	38.78	85.30

Table 3. **LeAP Class Per Sequence Evaluation.** We report evaluation of our camera branch label initialization, grouping classes and considering only labeled points, for each training sequence on the SemanticKITTI dataset.

Scene	Avg % Labelled	Overall Accuracy	Mean IoU	Weighted Mean IoU
00	50.61%	80.01%	25.71%	69.34%
01	51.59%	89.95%	26.47%	70.22%
02	48.19%	79.73%	24.89%	67.01%
03	43.79%	72.51%	25.76%	70.86%
04	57.23%	71.65%	28.33%	71.65%
05	45.33%	78.14%	25.58%	66.32%
06	40.96%	80.89%	21.21%	69.03%
07	48.45%	74.05%	25.94%	71.38%
09	50.20%	81.76%	24.65%	68.15%
10	36.86%	70.11%	25.87%	71.42%
Average	**47.32%**	**77.88%**	**25.44%**	**69.53%**
Average no Propagation	**10.01%**	**59.06%**	**13.01%**	**43.11%**
Improvement	**+372.9%**	**+31.9%**	**+95.5%**	**+61.3%**

Table 4. **Per-Scene Pseudo Labels Evaluation**. Evaluation of semantic pseudo labels against the ground truth labels from SemanticKITTI [2] training sequences. After reporting the average number of LiDAR points that are labelled in the single scans, accuracy is reported through mean Intersection over Union (IoU) and a weighted version of mean IoU (Weighted Mean IoU) that accounts for the number of labels per class in each scene. The presented results consider all the points and all classes: the generated pseudo labels, where available, can be very accurate, moreover on the classes with highest mapping in the dataset, as testified by the weighted IoU

Method	mIoU All ($\Delta\uparrow$)	Object $\Delta\uparrow$	Facility $\Delta\uparrow$
Lasermix [10] + 20% Proposed	60.87 **(+2.11)**	car, **+2.4**	building, **+4.1**
Redal [23] + 20% Proposed	61.2 (**+1.4**)	truck, **+2.9**	road, **+2.1**

Table 5. Distillation benefits (in **bold**) from geometry-guided labels produced by our method.

context, as mostly in further regions the measurements noise affects speed computation. We perform an evaluation carried out without applying the spatial consistency (*S.C.*) check and applying it enforcing a constraint of 5 moving neighbors. We specifically select for this evaluation sequences in which no object moves perpendicularly to ego vehicle resulting in 1000

samples, as the Doppler shift can only be measured along each ray. In practice, as the LiDARs cover a 360° coverage, sensors scan all direction around the vehicle and as objects have a none zero extend, failure in the ground-truth measurements can only be induced in sequences where vehicles drive on a circular trajectory around the ego vehicle. Final results are summarized in Table 6, where we compare against the predictions generated by 4DMOS [16] and find that our refined moving object segmentation improves accuracy in detecting moving points. In the same table we evaluate how different settings of the threshold τ for static separation (originally set to 0.5) affects the moving points detection.

Metric	**4DMOS [16] w/o S.C.**	**Ours w/o S.C.**	**4DMOS [16] w S.C.**	**Ours w S.C. & $\tau = 0.2$**	$\tau = 0.3$	$\tau = 0.7$
Accuracy	0.9803	0.9857	0.9811	**0.9874**	0.9791	0.9866
Recall	0.5916	0.6480	0.6624	**0.6848**	0.7910	0.6697

Table 6. **FMCW LiDAR Moving Object Segmentation.** We evaluate our moving object segmentation with the FMCW velocity measurements, with and without applying a spatial proximity filter for measurement consistency (*S.C.*), removing cluttered single points from the ground truth. We demonstrate the effectiveness of our moving object segmentation by comparing with pretrained 4DMOS [16].

In Table 7b we analyze how the same values of static separation τ_s (default 0.5) affect our pseudo bounding boxes and in Table 7a we explore how the propagation radius r affects semantic labels quality.

	Proposed $\mathbf{r = 0.2}$	$\mathbf{r = 0.1}$	$\mathbf{r = 0.3}$
mIoU [%]	**64.9**	60.7	48.3
Cat-mIoU [%]	**76.2**	70.4	55.3
Labelled [%]	66.9	61.5	76.8

(a) Propagation radius r on Semantic KITTI scene 08

	Proposed $\tau_\mathbf{s} = 0.5$	$\tau_\mathbf{s} = 0.3$	$\tau_\mathbf{S} = 0.7$
mAP [%]	**31.0**	25.8	28.3
NDS [%]	**45.2**	39.4	35.5

(b) Static separation threshold τ_s on our Long Range dataset

Table 7. **Ablation Studies.** (a) Influence of the propagation radius on point-wise semantic segmentation, evaluating on *val* sequence 08 of the SemanticKITTI dataset. (b) Impact of the static separation threshold τ_s on pseudo-box quality, evaluating on evaluation set of our Long Range dataset.

Table 8 shows the drop in performances of two NMRF models trained on our dense LiDAR scans ablated of our *Adaptive Spherical Occlusion Culling* and of f_{iwu} function and evaluated both on our consistent ground-truth and on the sparse LiDAR (where possible). Finally, Table 9 replicates the evaluation of Table 2 of the main paper using only sparse LiDAR measurements (for the available 0-80m range). This shows that the performance gains are not just a consequence of fine-tuning, as in every experiment NMRF is trained on the sparse LiDAR input.

	Pseudo	**Dense LiDAR Evaluation**						**Sparse LiDAR Evaluation**					
		MAE ↓ [m]			**RMSE ↓ [m]**			**MAE ↓ [m]**			**RMSE ↓ [m]**		
		0-80	80-150	150-250	0-80	80-150	150-250	0-80	80-150	150-250	0-80	80-150	150-250
KITTI	Oracle	4.48	22.03	30.76	7.62	25.66	35.83	4.24	-	-	5.79	-	-
	Proposed	**3.28**	**9.57**	**17.43**	**5.66**	**13.49**	**21.89**	**3.26**	-	-	**5.67**	-	-
	Proposed without ASOC	4.49	12.72	18.09	7.86	16.23	22.48	4.22	-	-	7.81	-	-
	Proposed without f_{iwu}	3.91	10.04	17.50	6.22	15.05	21.99	3.73	-	-	5.99		
LR	Oracle	5.44	20.79	31.83	7.98	25.70	38.96	5.08	19.32	31.03	8.87	26.13	39.75
	Proposed	**2.27**	**6.14**	**21.07**	**4.21**	**9.81**	**25.16**	**2.48**	**7.02**	20.80	**4.91**	**9.67**	**24.14**
	Proposed without ASOC	3.39	8.12	22.47	6.68	12.93	26.91	3.56	8.19	21.51	6.13	12.14	25.48
	Proposed without f_{iwu}	3.58	10.40	22.91	6.06	14.89	26.64	3.70	9.13	**20.01**	6.05	12.23	24.72

Table 8. **Depth Estimation Ablation**. We train 2 NMRF [8] models with projected dense LiDAR, ablated of our *Adaptive Spherical Occlusion Culling* (ASOC) and of our f_{iwu} function, highlighting the importance of these component in order to precisely recover occlusions and remove ghost artifacts for depth estimation. We evaluate on KITTI and on our Long Range (LR) datasets, both using our dense consistent LiDAR and sparse LiDAR for evaluation.

	Metric	Oracle	LIO-SAM	CREStereo	DA-V2	Ours
KI	MAE	4.2	5.7	5.3	5.9	**3.3**
	RMSE	5.8	9.6	6.6	7.7	**5.7**
LR	MAE	5.1	13.4	11.5	12.7	**2.5**
	RMSE	8.9	17.1	14.3	16.5	**4.9**

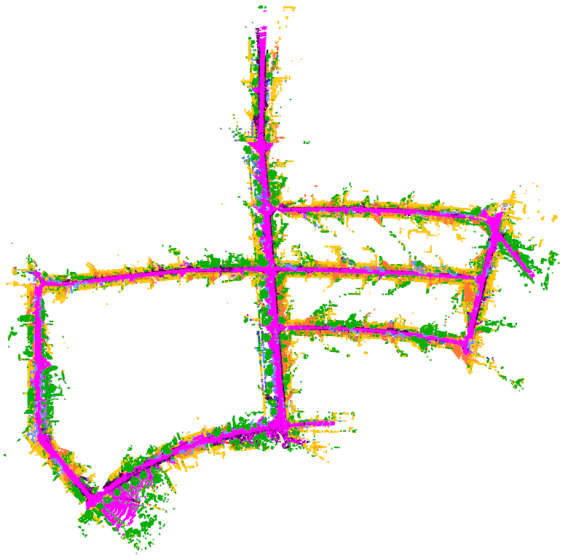
Table 9. **Sparse LiDAR Evaluation.** Evaluation of NMRF using the sparse LiDAR for KITTI (KI) and our long range (LR) dataset, in the available 0-80 meters range. For Oracle and Proposed, more ranges are evaluated in the previous Table 8.

6. Additional Results

In this section we provide additional visual results of the outputs of our method.
Figure 6, shows examples from SemanticKITTI [2] scenes of fully pseudo labeled maps.
Figure 7, collects evidences that augmenting ground truth data with our pseudo labels is beneficial for recovering oracle performances: we present predictions on validation sequence 08 and highlight in red the discrepancies with the ground truth labels.
Figure 8 shows some comparison between refined and unrefined maps, showcasing how the initial moving object estimate has minimal impact on performance, as our *Iterative Weighted Update Function* f_{IWU} can recover from errors, even in scenes with many high-speed actors and artifact (Scene 3).
In Figures 9, 10 , 11 we show many examples of our accumulated depth-maps for the SemanticKITTI dataset, projecting them into the camera frame and highlighting the range and density increase respect to the sparse LiDAR, in diverse ranges and with diverse dynamic actors.
Finally Figures 12 13 present a collection of our pseudo-bounding boxes for different ranges and object sizes.

(a) Scene 00

(b) Scene 02



(c) Scene 04

(d) Scene 08

Figure 6. **Fully Labeled KITTI Map**. Visualization of labeled map points from scene 00, 02, 04 and 08 in the Semantic KITTI dataset [2], after applying our algorithmic label propagation.

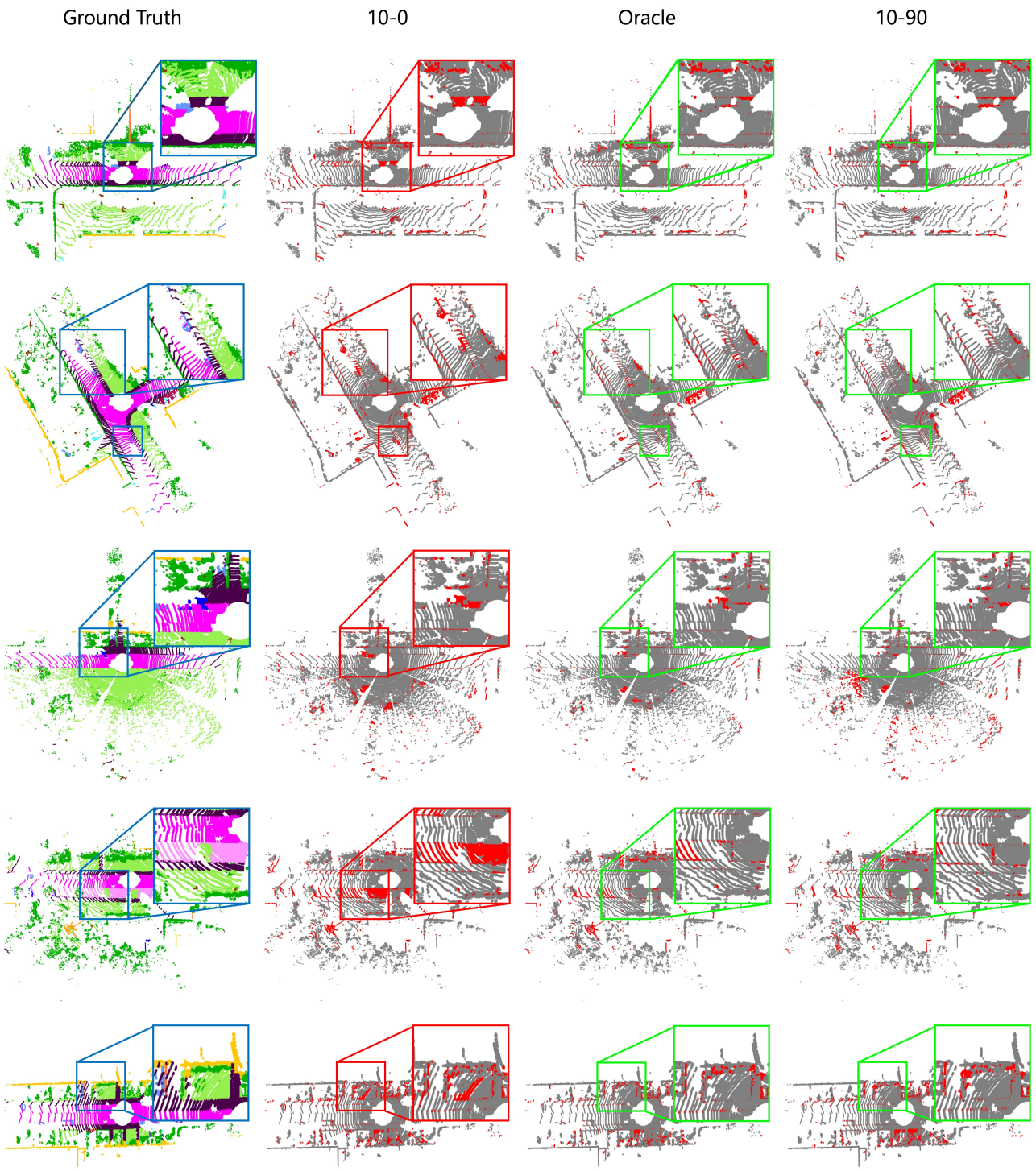


Figure 7. **Additional Qualitative Results for Semantic Segmentation.** We present additional results of predictions of the models trained respectively on 10% ground truth labels (*10-0*), on 100% ground truth labels (*Oracle*), and on 10% ground truth labels coupled with 90% of our pseudo labels. In red are highlighted all points where prediction label is different from ground truth one. As shown in the main manuscript, the addition of our pseudo labels let the model achieve near-Oracle quality, confirming that our method can generate a valid additional ground truth for unlabeled datasets.

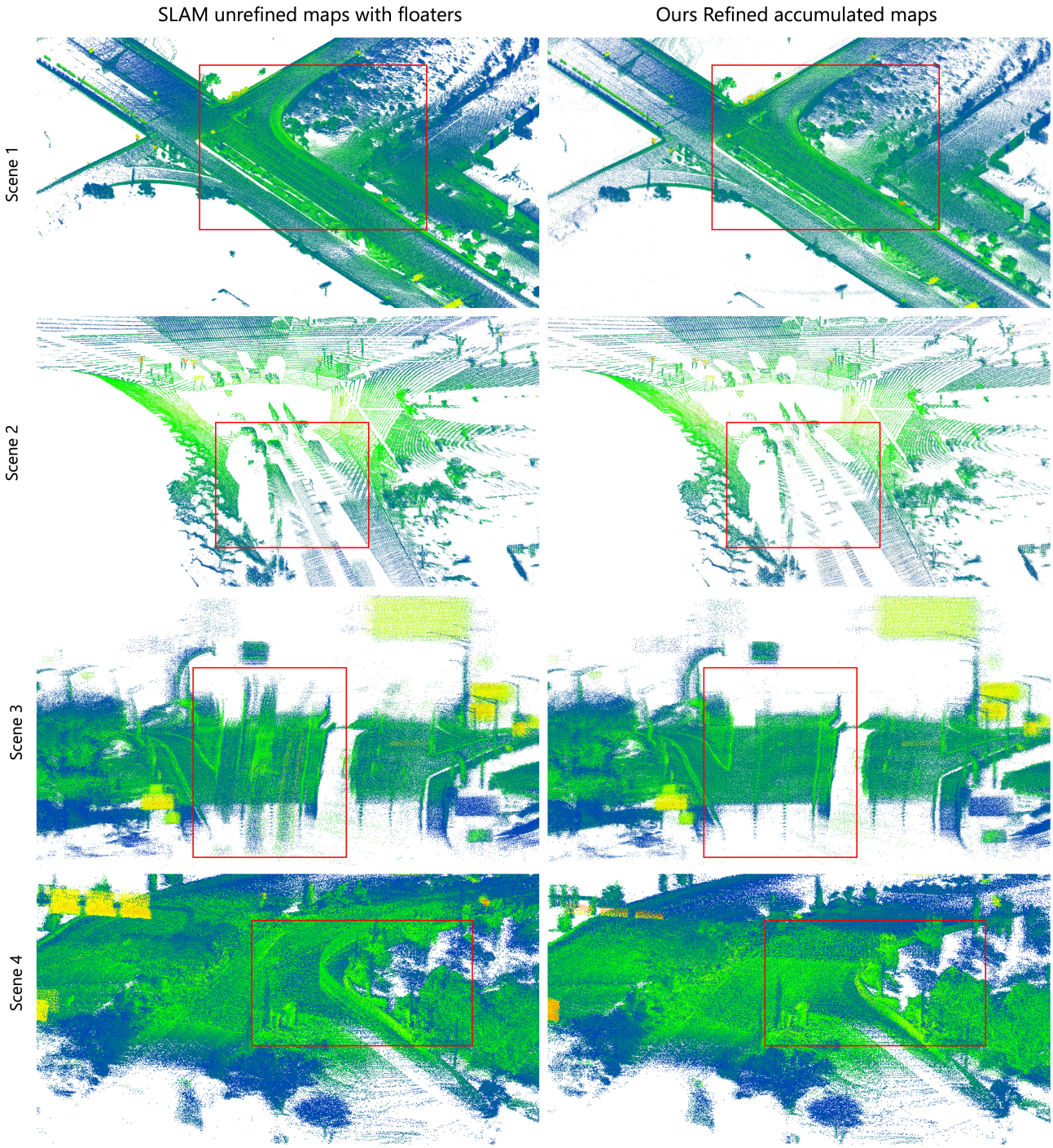


Figure 8. **Floaters Removal Stage.** This figure illustrates the effectivity of removing floating artifacts in our pipeline from moving objects for four scenes (rows). Our approach is able to remove consistently floaters (on the right) respect to the normal SLAM (on the left), in which points belonging to moving objects are retained in the map and appear smeared along their trajectories.

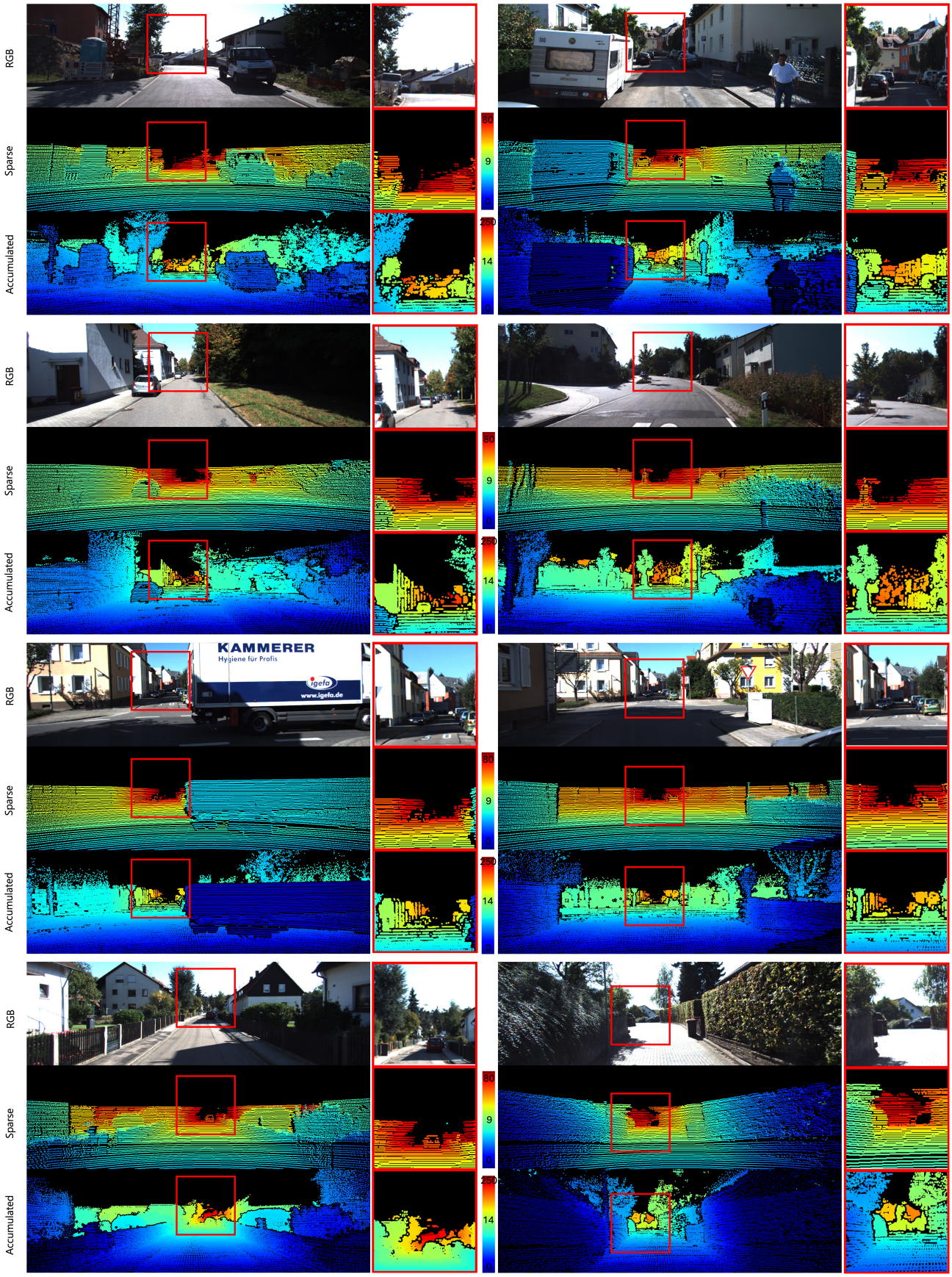


Figure 9. **High Quality Long Range Depth Maps.** Our method is capable of overcoming the limited perception distance of the sparse LiDAR both in static and dynamic conditions.

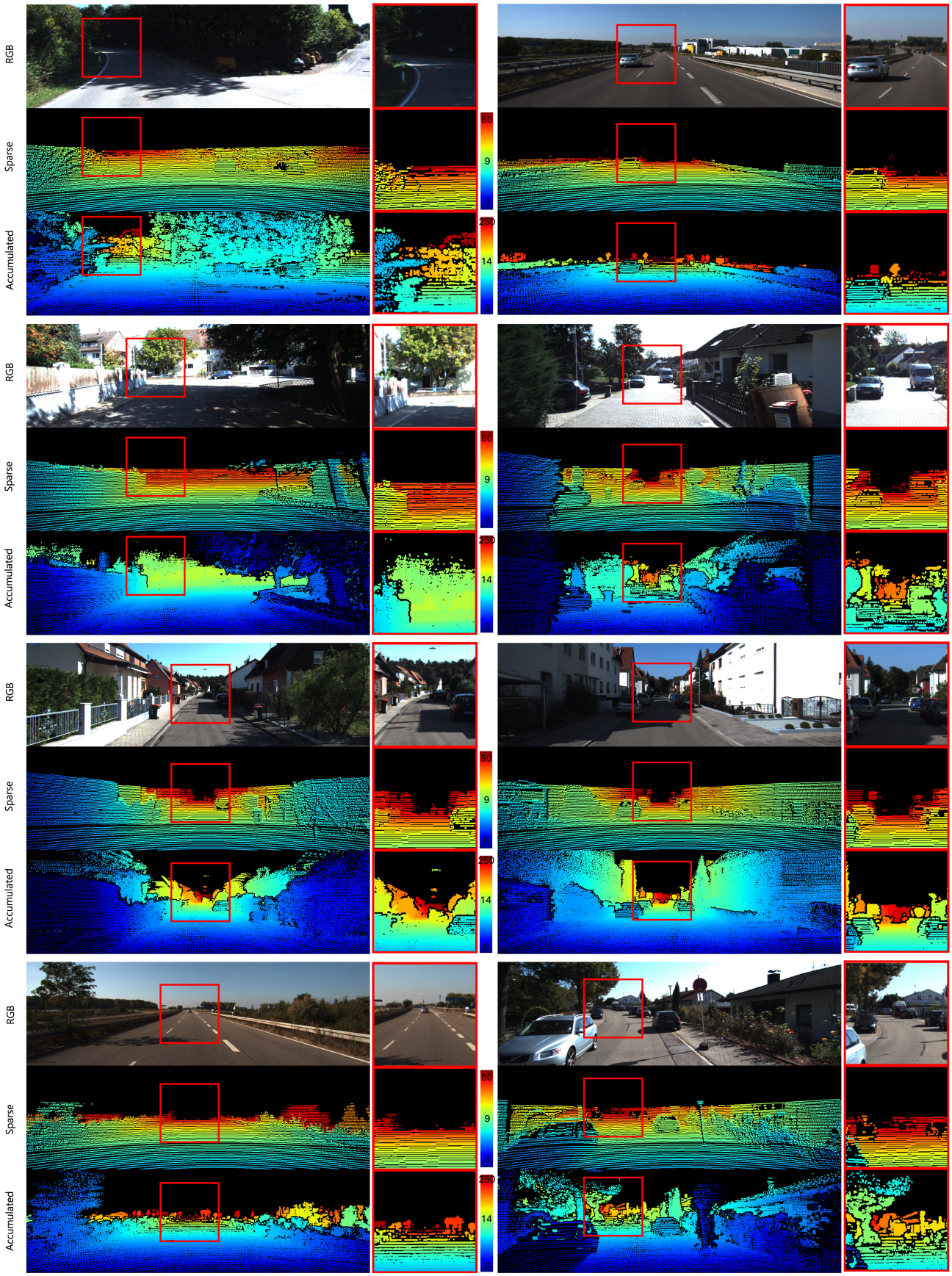


Figure 10. **High Quality Long Range Depth Maps.** Our method can uncover areas that normally are not detected, increasing density and providing accurate long range depth cues.

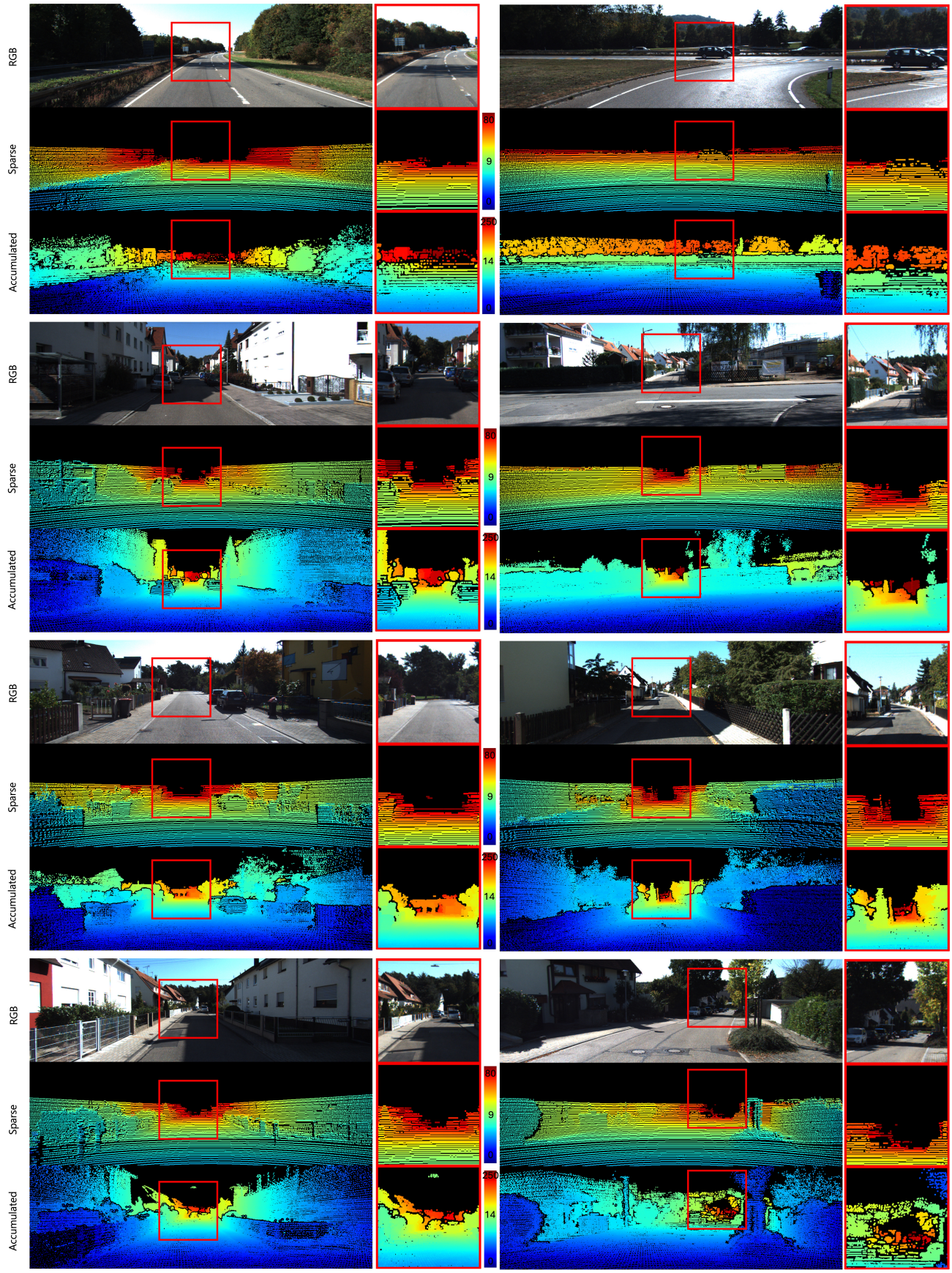


Figure 11. **High Quality Long Range Depth Maps.** Our method can uncover areas that normally are not detected, increasing density and providing accurate long range depth cues.

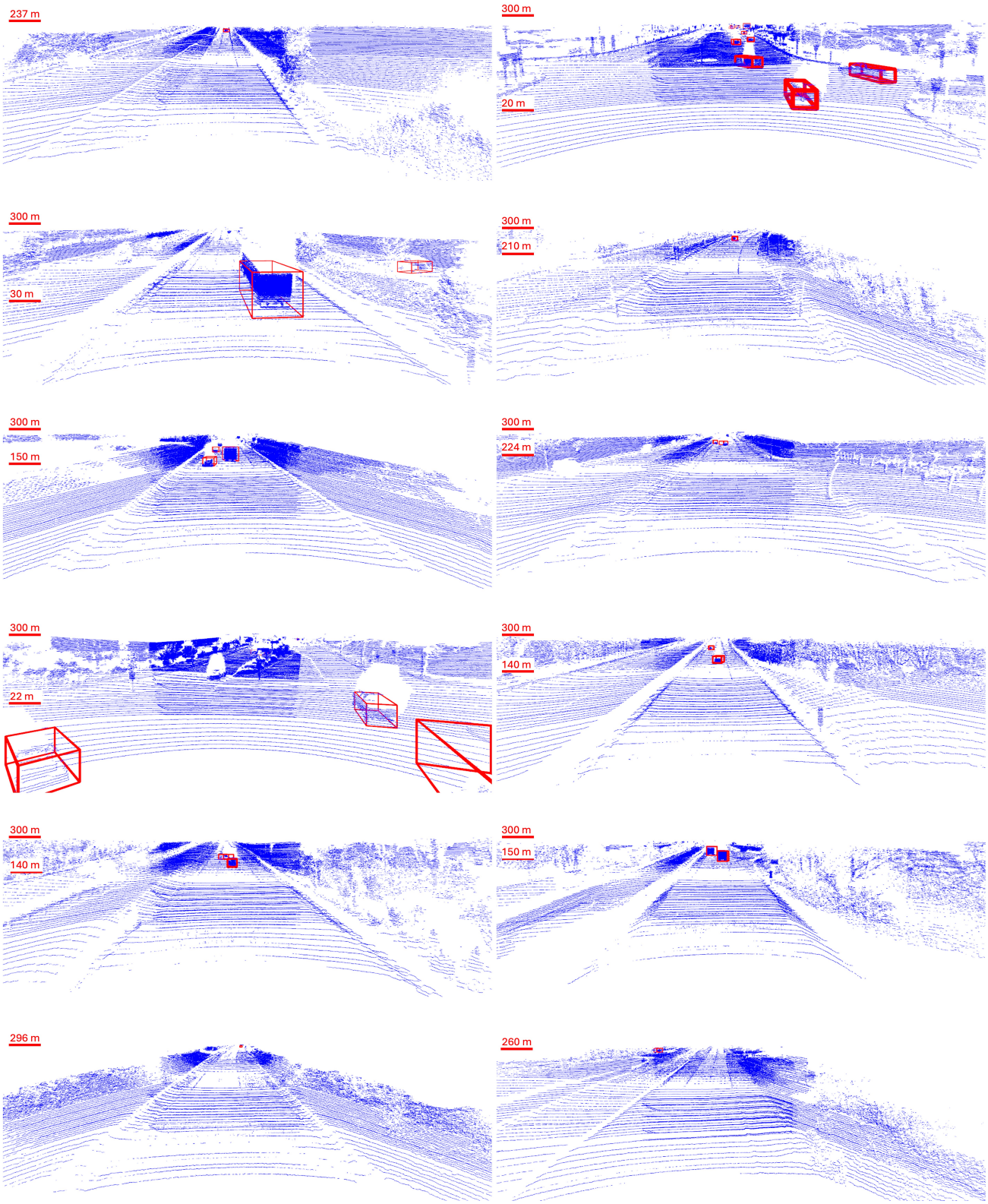


Figure 12. **Pseudo Bounding Boxes.** We present some examples of our pseudo bounding boxes at different ranges.

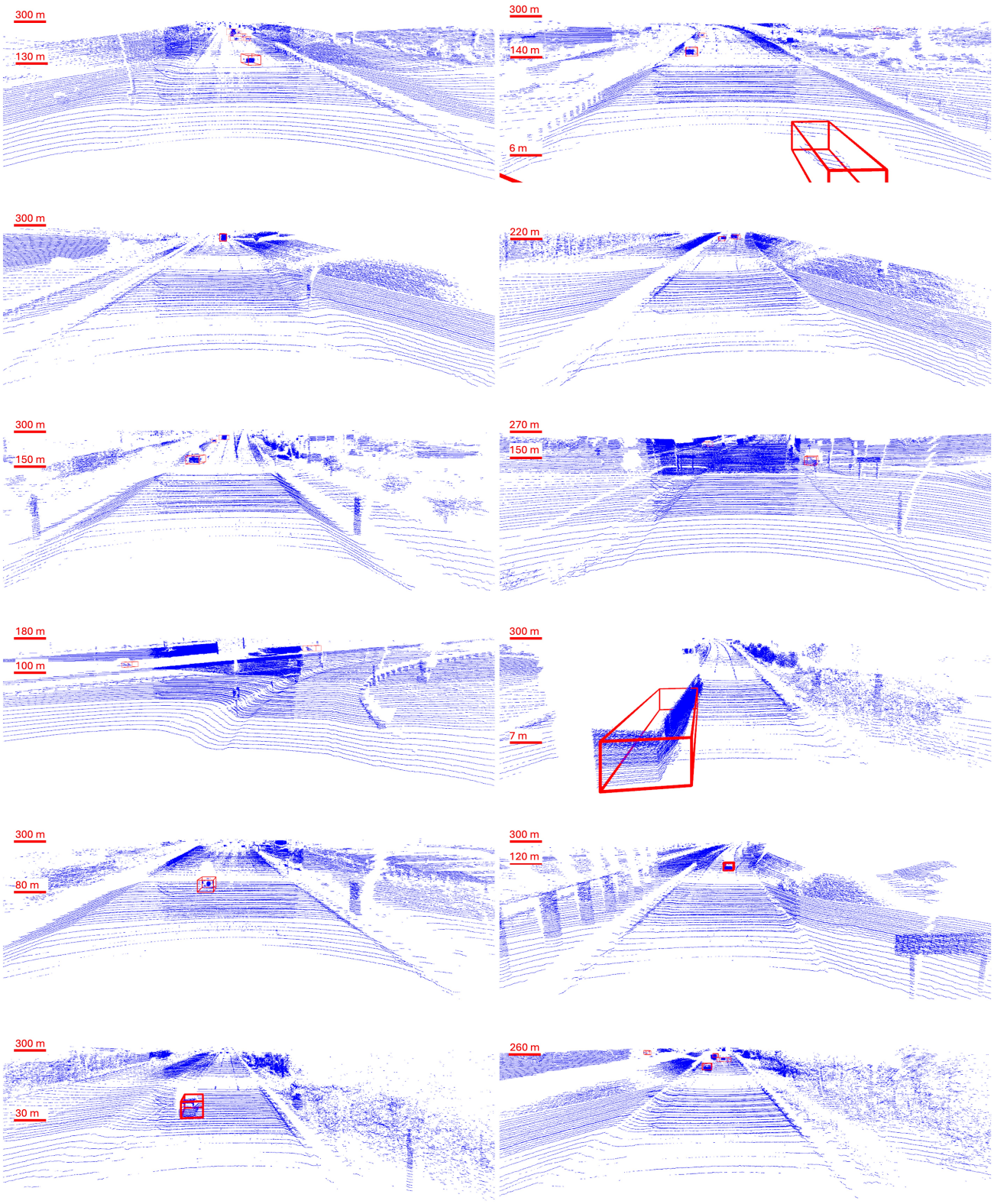


Figure 13. **Pseudo Bounding Boxes.** We present some examples of our pseudo bounding boxes at different ranges.

References

[1] Stefan Baur, Frank Moosmann, and Andreas Geiger. Liso: Lidar-only self-supervised 3d object detection. In *European Conference on Computer Vision (ECCV)*, 2024. 9

[2] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*, 2019. 1, 5, 9, 11, 13, 14

[3] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3, 9

[4] Jiaqi Chen, Zeyu Yang, and Li Zhang. Semantic segment anything. https://github.com/fudan-zvg/Semantic-Segment-Anything, 2023. 9

[5] Simone Ferrari, Luca Di Giammarino, Leonardo Brizi, and Giorgio Grisetti. Mad-icp: It is all about matching data–robust and informed lidar odometry. *IEEE Robotics and Automation Letters*, 2024. 4

[6] Simon Gebraad, Andras Palffy, and Holger Caesar. Leap: Consistent multi-domain 3d labeling using foundation models, 2025. 9, 10

[7] 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, 2012. 3

[8] Tongfan Guan, Chen Wang, and Yun-Hui Liu. Neural markov random field for stereo matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5459–5469, 2024. 8, 12

[9] Yuenan Hou, Xinge Zhu, Yuexin Ma, Chen Change Loy, and Yikang Li. Point-to-voxel knowledge distillation for lidar semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 8479–8488, 2022. 9, 10

[10] Lingdong Kong, Jiawei Ren, Liang Pan, and Ziwei Liu. Lasermix for semi-supervised lidar semantic segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21705–21715, 2023. 9, 11

[11] 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

[12] 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 Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019. 9

[13] Jiankun Li, Peisen Wang, Pengfei Xiong, Tao Cai, Ziwei Yan, Lei Yang, Jiangyu Liu, Haoqiang Fan, and Shuaicheng Liu. Practical stereo matching via cascaded recurrent network with adaptive correlation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16263–16272, 2022. 8

[14] Yiyi Liao, Jun Xie, and Andreas Geiger. Kitti-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3292–3310, 2023. 9

[15] Yancong Lin and Holger Caesar. Icp-flow: Lidar scene flow estimation with icp. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15501–15511, 2024. 9

[16] 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. 4, 12

[17] Y. Pan, X. Zhong, L. Wiesmann, T. Posewsky, J. Behley, and C. Stachniss. Pin-slam: Lidar slam using a point-based implicit neural representation for achieving global map consistency. *IEEE Transactions on Robotics (TRO)*, 40, 2024. 4

[18] 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. 4, 5, 8, 9

[19] 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, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3

[20] Ignacio Vizzo, Tiziano Guadagnino, Benedikt Mersch, Louis Wiesmann, Jens Behley, and Cyrill Stachniss. KISS-ICP: In Defense of Point-to-Point ICP – Simple, Accurate, and Robust Registration If Done the Right Way. *IEEE Robotics and Automation Letters (RA-L)*, 8(2):1029–1036, 2023. 9

[21] Qitai Wang, Yuntao Chen, Ziqi Pang, Naiyan Wang, and Zhaoxiang Zhang. Immortal tracker: Tracklet never dies. *arXiv preprint arXiv:2111.13672*, 2021. 6

[22] Xinshuo Weng, Jianren Wang, David Held, and Kris Kitani. AB3DMOT: A Baseline for 3D Multi-Object Tracking and New Evaluation Metrics. *ECCVW*, 2020. 6

[23] Tsung-Han Wu, Yueh-Cheng Liu, Yu-Kai Huang, Hsin-Ying Lee, Hung-Ting Su, Ping-Chia Huang, and Winston H Hsu. Redal: Region-based and diversity-aware active learning for point cloud semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15510–15519, 2021. 9, 11

[24] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. *arXiv:2406.09414*, 2024. 9

[25] Xinge Zhu, Hui Zhou, Tai Wang, Fangzhou Hong, Wei Li, Yuexin Ma, Hongsheng Li, Ruigang Yang, and Dahua Lin. Cylindrical and asymmetrical 3d convolution networks for lidar-based perception, 2021. 9, 10

[26] Laurent Zwald and Sophie Lambert-Lacroix. The berhu penalty and the grouped effect, 2012. 8