TruckDrive: Long-Range Autonomous Highway Driving Dataset (Supplementary Information)

Filippo Ghilotti[1] Edoardo Palladin[1] Samuel Brucker[1] Adam Sigal[1]
Mario Bijelic[1,2] Felix Heide[1,2]
[1]Torc Robotics [2]Princeton University

This document provides information in support of the findings from the main manuscript. In Sec. 1, we report additional details about the sensor setup used to record TruckDrive. In Sec. 2, we include additional visualization of diverse samples contained in the proposed dataset. In Sec. 3, we report additional details regarding our labeling and refinement procedures with additional statistics. In Sec. 4, we provide additional details on how we generate the moving object segmentation ground truth and the accumulated LiDAR ground truth to evaluate all depth estimation baselines. In Sec. 5, we detail all design choices, training configurations and pre-processing needed to reproduce the reported baselines experiments from the main manuscript. We provide additional qualitative visualization and results in the attached video.

All authors contributed equally to this work.

Contents

1. Additional Dataset Details

In this section, we expand on three aspects of TruckDrive. First, we provide additional details about the sensor setup, including camera count and frequency clarifications. Second, we provide more context about the dataset domain and how we derived the scene taxonomy using a pretrained vision–language model. Finally, we report more details about the recordings, including how we performed frame counts and scene length measurements.

1.1. Sensor Setup Details

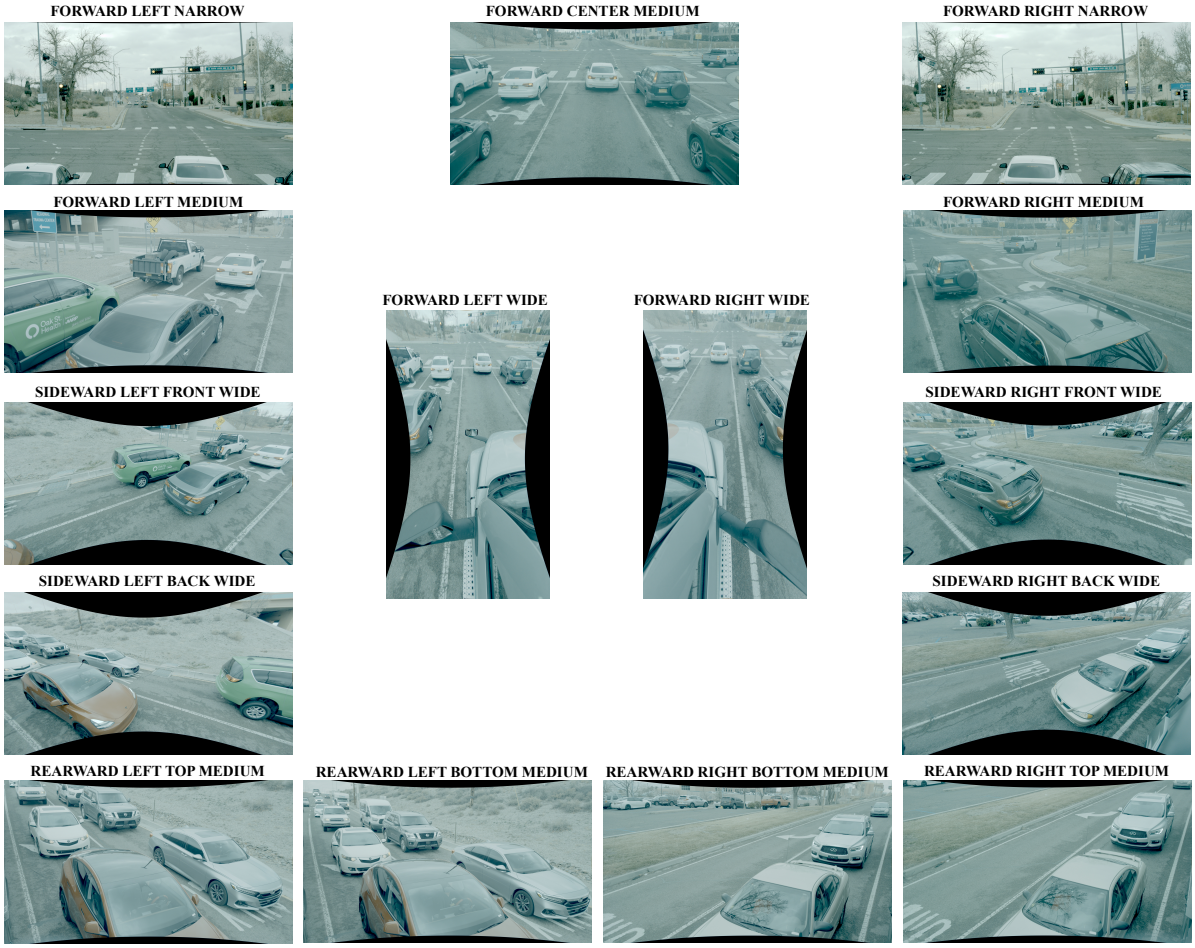


Figure 1. **Camera Rig Configuration.** Full 15 cameras configuration, highlighting their placement (forward, sideward, rearward) and field of view (narrow, medium, wide). This differs from the 11 camera rig due to the presence of additional forward left/right medium cameras and rearward left/right top medium cameras.

4D Long Range LiDAR System Our primary $3D$ capturing is performed by a suite of seven LiDARs, specifically the AEVA Aeries II, which jointly provide a dense, 360 degrees field of view, operating at a 10 Hz scan rate with an effective detection range of up to 400 meters. Each LiDAR provides $\sim$ 100 lines of resolution and a Field of View (FOV) of $120° \times 30°$. A key feature of these sensors is the use of Frequency-Modulated Continuous-Wave (FMCW) technology, which allows to capture instantaneous radial velocity v_r for each point in the point cloud. The velocity measurement is derived from the Doppler-induced phase shift $\Delta\phi$ through

$$v_r = \frac{\Delta\phi \cdot \lambda}{4\pi} \cos\theta \tag{1}$$

where λ is the laser's wavelength and θ is the angle of incidence.

Table 1. **Sensor Specifications and Raw Data Scale.** We present in detail our sensor platform, including RCCB cameras 3D short-range (SR) LiDARs, a 4D long-range (LR) FMCW LiDAR, and 4D radars, capturing 475 thousands synchronized frames

	Camera	**LiDAR**		**Radar**
	RCCB *AR0820*	**4D LR** *Aeries II*	**3D SR** *OS0/OS1*	**4D** *ARS540*
Make	OnSemi	AEVA	Ouster	Continental
Type	RCCB	FMCW 4D	3D	FMCW 4D
Resolution	3848×2168	$\sim$100 lines	$64/128 \times 2048$	—
FOV (H$\times$V)	$52.8° \times 28.9°$	$120° \times 30°$	$360° \times 90°/45°$	$\pm 4°$–$\pm 20°$
f **(Hz)**	5–10	10		20
Raw Captures	6.3M	7.8M		6.0M
Sync Timestamps	569k	744k		601k
Cross-Modal Sync Timestamps: 475k				

Additionally, it relies on coherent optical interference and can often suppress continuous weather scattering and "see through" clutter over a much longer instrumented range (500 m), not requiring differentiation of multiple return peaks to maintain long-range visibility

3D Short Range Surround LiDAR System To ensure a comprehensive and dense point cloud in the near-field, our sensor suite includes 3 short-range LiDARs: one Ouster OS1 and two Ouster OS0. These sensors provide a complete $360°$ horizontal field of view and are strategically positioned to mitigate the typical blind spots observed in large semi-trucks namely, the regions directly in front of the cab, immediately behind the trailer, and along the lateral sides adjacent to the doors and trailer edges. The OS1 offers reliable mid-range sensing, with a resolution of 128×2048 and a field of view of $360° \times 45°$, while the OS0 offers a resolution of 64×2048 and features an ultra-wide vertical field of view of $360° \times 90°$, making it particularly effective at detecting low-lying objects and road hazards close to the vehicle.

Imaging System. The visual data is captured by a suite of *OnSemi AR0820* cameras, each featuring a $1/2$-inch CMOS sensor. This camera array is positioned and synchronized to provide a near-360 degrees field of view, operating at a frequency of $5 - 10$ Hz. Each sensor provides a high-resolution image of 3848×2168 pixels (8.3 MP). The suite comprises 3 different focal lenses camera: narrow ones, with focal length of roughly 7350 pixels, medium ones, with focal length of roughly 3304 pixels, and wide ones, with focal length of roughly 1330 pixels. The sensors output raw data in a *RCCB* (Red-Clear-Clear-Blue) color filter format, which is optimized for low-light performance and high dynamic range with three exposures and 21 bit depth. Images are subsequently undistorted and a custom tone mapping, tailored to autonomous driving computer vision application, is applied.

Camera Subsets. In the main manuscript we report a range of 11 to 15 cameras because all sequences use an 11-camera surround-view rig, and a subset of scenes includes two additional pairs. The complete 15-camera configuration, shown in Figure 1, augments the base rig with a front-facing medium-focal-length pair with partially overlapping field of view (forward left/right medium cameras) and a rearward-facing medium-focal-length pair with no overlap (rearward left/right top medium). These views can be treated as held-out targets for neural rendering, enabling a rigorous evaluation of novel view synthesis performance. At the same time, the two rearward top–bottom cameras form a vertical stereo pair, adding backward-looking stereo capabilities that are particularly valuable for long-range depth cues. Finally, this extended configuration allows us to investigate view augmentation strategies during training, where additional viewpoints can be used to supervise and stress-test the network's ability to extrapolate beyond its nominal input distribution. Overall, 2,647 out of 3,828 scenes are captured with the full 15-camera rig.

Camera Framerate. In the main manuscript we report camera capture frequencies of 5 and 10 Hz. Specifically, 2,647 out of 3,828 scenes are recorded with cameras running at 5 Hz, while the remaining scenes use a higher camera frame rate of 10 Hz. The upsampled portion of the dataset is compatible with the lower frequencies by skipping in between synchronized samples.

4D Radar Capturing System. To complement our LiDAR and camera systems, we have integrated a suite of 10 Continental ARS 540 radar sensors. These sensors are distributed around the vehicle to provide full 360 degrees coverage. The primary advantage of the radar system is its exceptional performance in adverse weather conditions and the rich point cloud detections completed with precise velocity information for each point, adding a critical layer of redundancy and enriching the dataset

with reliable dynamic information.

Geo-Inertial Measurements and Ground Truth Poses. Our semi-truck is equipped with 2 GNSS and 4 IMUs for precise geo-localization and motion estimation. We additionally fuse position and velocity measurements from synchronized captures in a tailored Post Processing Kinematic (PPK) pipeline to obtain highly reliable ego poses and complement them with LiDAR SLAM [37] to generate ground truth poses useful to train planning algorithms through imitation learning or localization algorithms in future research papers..

1.2. Dataset Domain

Our data collection campaign covers various highways and scenarios from west, Arizona, to east, Virginia, spanning 8 U.S. states (NM, TX, VA, NC, TN, AR, WV, AZ). In Figure 2 we show the Highway net of the US, highlighting the main ones traversed by our semi-truck during our data collection campaign.

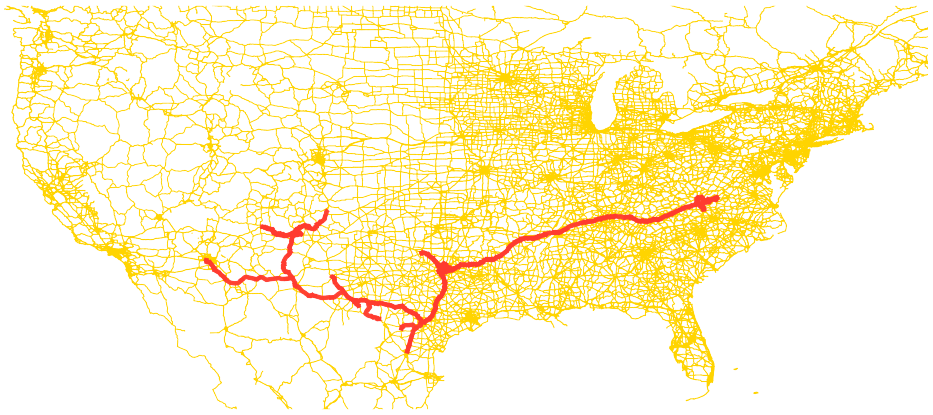


Figure 2. **TruckDrive Data Collection Campaign.** We highlight the highways traversed by our semi-truck in order to collect 3828 diverse scene, spanning from West to East of the United States.

Scene Taxonomy. To obtain a coarse taxonomy of driving conditions, we use the vision-language model Qwen3-VL-30B-Instruct [50] to automatically classify each scene from a single representative camera view. For every scene, we sample one RGB image and query Qwen3 with a structured prompt asking for: a free-form textual description, a discrete scene type in $\{$*highway*, *urban*, *extra-urban*$\}$ and a time-of-day label in $\{$*day*, *night*, *dawn*, *dusk*$\}$. We run Qwen3 via vLLM with deterministic decoding (temperature $= 0$) and store the resulting structured text, from which we aggregate counts of highway vs. urban vs. extra-urban scenes and the distribution of time-of-day categories reported in the main manuscript.

1.3. Scene Recording Information

Frame Count. In the main manuscript we report a total of 475 thousand frames with 165 thousands annotated frames. A single *frame* always refers to a fully synchronized multi-sensor snapshot, not a single image or LiDAR scan. Concretely, each frame corresponds to one global timestamp at which we provide up to 15 camera images, 7 long-range LiDAR point clouds, 3 short-range LiDAR point clouds and 10 radar point clouds, all temporally aligned within a tight synchronization window, as described in the main manuscript. Annotations are defined at this frame level, so every annotated frame includes labels that are consistent across all sensors at that timestamp, with tracking IDs for temporal consistency. To facilitate this data handling, in addition to per-sensor recordings, we provide, for each frame, a fused LiDAR scan obtained by transforming and merging all 7 long range LiDAR point clouds into the forward LiDAR reference frame and an analogous fused radar point cloud where all radar detections are projected into the forward radar frame.

Scenes Length. In the main manuscript we report an average of 20 seconds for scene length. In particular 2094 scenes are 15 seconds long, 308 are 20 seconds long and the remaining 1426 are 25 seconds long.

Table 2. **Detailed Label Mapping and Sizes.** We present the full mapping from coarse evaluation categories to fine-grained labels, and compute statistics about average 3D box sizes (ℓ, w, h) per category.

Category	**Avg box size** (ℓ, w, h) $[m]$	**Fine-grained labels**
DontCare	-	DelineatorGroupDontCare, OutOfLidarRangeVehicleGroup, ParkingLotVehicleGroup, Vehicle-EgoVehicle-Cab, Vehicle-EgoVehicle-Trailer
Two-Wheel	(1.59, 1.14, 0.71)	VRUvehicle-Bicycle, VRUvehicle-Motorcycle, VRUvehicle-StandingScooter, VRUvehicle-Wheelchair, Vehicle-Bicycle, Vehicle-Motorcycle
Passenger-Car	(4.82, 1.70, 1.89)	Vehicle-Passenger
Person	(0.62, 1.67, 0.70)	Animal, Person, Person-Other, Person-Pedestrian, Person-Rider, Person-Skater, Person-TrafficControl
RoadObstruction	(0.41, 1.04, 0.41)	RoadDebris, RoadDebris-Other, RoadDebris-Pothole, RoadDebris-RoadKill, RoadDebris-Tire, RoadDebris-Vegetation, RoadObstruction, RoadObstruction-Barrel, RoadObstruction-Barricade, RoadObstruction-Cone, RoadObstruction-Delineator, RoadObstruction-Other, RoadObstruction-VerticalPanel
SemiTruck-Cab	(5.52, 3.62, 2.92)	Vehicle-SemiTruck-Cab
SemiTruck-Trailer	(15.04, 3.72, 2.90)	Vehicle-SemiTruck-Trailer
Vehicle	(6.05, 2.07, 2.12)	VRUvehicle, VRUvehicle-Other, VRUvehicle-Trailer, Vehicle, Vehicle-Bus, Vehicle-DeliveryVan, Vehicle-Equipment, Vehicle-HeavyVehicle, Vehicle-Other, Vehicle-RV, Vehicle-SchoolBus, Vehicle-SingleUnitTruck, Vehicle-Trailer, Vehicle-Unibody
TrafficSign	(0.11, 1.58, 1.26)	DynamicTrafficSign, GeneralTrafficSign, GeneralTrafficSign-Regulatory, GeneralTrafficSign-School, LaneUseSignal, LaneUseSignal-Green, LaneUseSignal-Off, LaneUseSignal-Red, TrafficSign, TrafficSign-Advisory, TrafficSign-Informational, TrafficSign-LaneEnds-Right, TrafficSign-Merge, TrafficSign-Merge-Right, TrafficSign-Other, TrafficSign-RoadworkAhead, TrafficSign-SpeedLimit, TrafficSign-SpeedLimit-Exit, TrafficSign-SpeedLimit-HardLimit, TrafficSign-SpeedLimit-Upcoming, TrafficSign-Yield, TrafficSignal, TrafficSignal-GreenForwardArrow, TrafficSignal-GreenLeftArrow, TrafficSignal-GreenRightArrow, TrafficSignal-GreenSolid, TrafficSignal-MultiState, TrafficSignal-Off, TrafficSignal-Other, TrafficSignal-RedLeftArrow, TrafficSignal-RedRightArrow, TrafficSignal-RedSolid, TrafficSignal-YellowLeftArrow, TrafficSignal-YellowRightArrow, TrafficSignal-YellowSolid, WalkSignal
EmergencyVehicle	(5.77, 2.16, 2.15)	Vehicle-Emergency, Vehicle-Police

2. Dataset Sample Visualizations

We illustrate the diversity of lighting and weather with representative dataset samples in TruckDrive in Figure 3, with examples from different times of day, illumination levels and adverse conditions. In Figure 4, we show representative road types, including urban segments, highways, and intersections, together with typical driving maneuvers such as merging, lane changes, and overtakes. For each example, we also provide videos that include the full recording with synchronized sensor captures and annotations. Finally, in Figure 5 we showcase some segments with non-nominal driving conditions, like urban scenes with jaywalking pedestrians, stopped or unusual vehicle types, and construction sites.

3. Labeling Details

We list the full fine-grained labels used in human annotation and how we grouped them in the final 9 coarse categories in Table 2. We group fine-grained labels into coarser categories to balance the number of instances per class and encourage appearance similarity within each group. As a design choice, we exclude the ego vehicle cabin (*Vehicle-EgoVehicle-Cab*) and trailer (*Vehicle-EgoVehicle-Trailer*), since we observed models overfitting to this unique configuration and yielding artificially inflated scores, especially for object detection. Vehicle subclasses are grouped so as to avoid excessive variance in physical size, although vehicles still remain the category with the highest intra-class variability; finer splits would result in classes with too few examples to be statistically meaningful. For *EmergencyVehicle*, we include all vehicle types that can significantly alter nominal planning behavior for human or autonomous drivers. This is complemented by fine-grained traffic sign annotations, which can support higher-level traffic understanding in the presence of disruptions or special operating conditions.

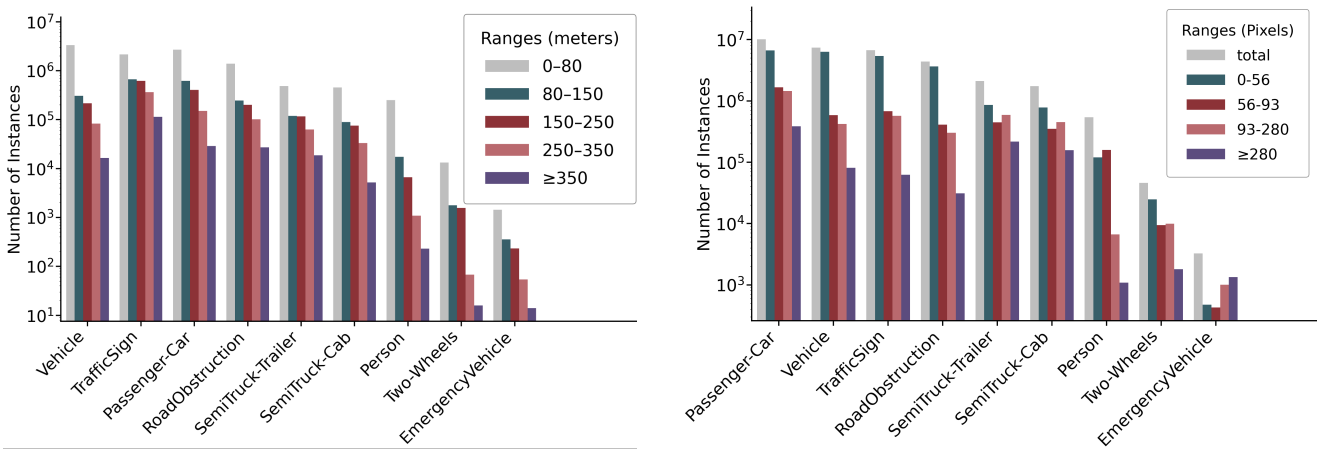
3.1. Dataset Instances Statistics

We present additional instance statics of TruckDrive dataset. In Table 2 we introduce average instance sizes for coarse label groups. In Figure 6 we present the class labels range distribution for 3D and 2D labels, showcasing the wide distribution of agents distance from the ego in the 3D setting and the distribution of 2D boxes dimension in pixel space. In Figure 7 we report the number of appearance of the fine-grained annotated classes in both 3D and 2D. We envision that the fine-grained labels can be useful to train planning or reasoning models to support higher-level traffic understanding.

Figure 3. **Light and Weather Variety Examples.** We show qualitative examples of the different light and weather conditions in our dataset. Samples span different times of day, leading to varied illumination, and include multiple weather types that can degrade perception performance and affect planning behavior.

Figure 4. **Driving Behaviour Variety Examples.** We show qualitative examples of different road types and driving behaviors. We couple these frames with full scene videos, with full camera setup, LiDAR, radar and bounding boxes in the supplemetary material provided.

Figure 5. **Non-Nominal Driving Examples.** We show qualitative examples of different non nominal driving conditions, like pedestrians on the road, construction sites, unusual vehicles types and stopped vehicles on the side of the road.



(a) 3D Class Labels Distance Range Distribution

(b) 2D Class Labels Pixel Dimension Range Distribution

Figure 6. **Dataset 3D-2D Label distribution.** We present per class range distribution and pixel dimension distribution for the 3D boxes (left) and 2D boxes (right) in TruckDrive.

3.2. Labeling Refinement

We present a schematic explanation of our labeling refinement procedure introduced in the main manuscript in Figure 8.

3.3. Odometry

To obtain accurate ego-motion for our driving dataset, we rely primarily on a high-precision Post-Processing Kinematic (PPK) pipeline. We fuse measurements from two GNSS receivers and four IMUs, which provides globally consistent poses for all synchronized frames. For all scenes, we additionally run two LiDAR-based odometry/SLAM methods:

- PIN-SLAM [37]: a neural implicit-map SLAM system that incrementally learns a local signed-distance representation and estimates poses via correspondence-free alignment to the learned map.
- MAD-ICP [13]: a robust, general-purpose LiDAR odometry method based on an ICP framework that uses PCA-augmented KD-trees and uncertainty-aware local map management.

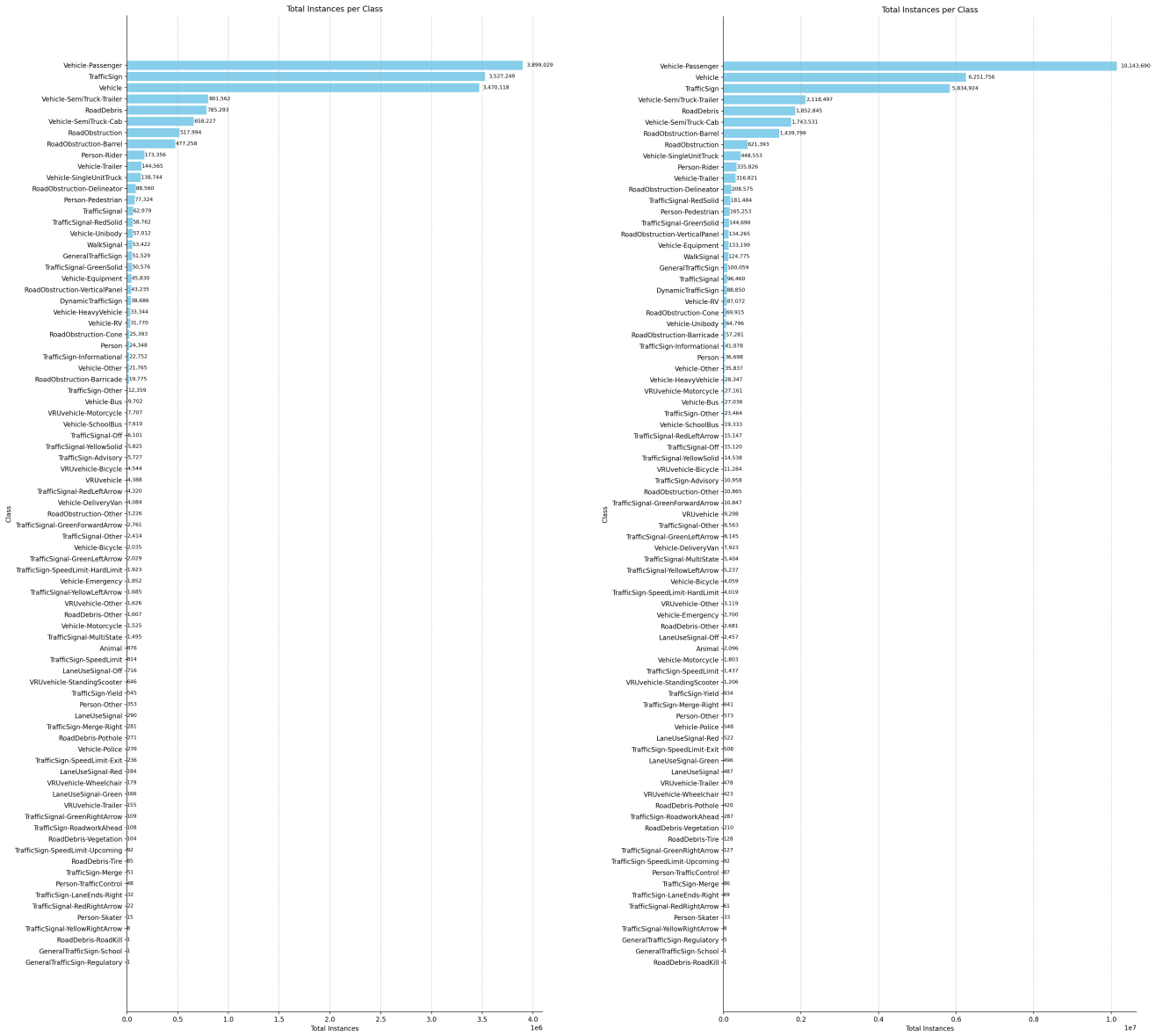
LiDAR odometry methods are particularly useful for tasks such as LiDAR accumulation because they optimize directly on geometric measurements. This allows them to filter out centimeter-level local irregularities (e.g., road bumps) that can still be present in PPK trajectories. Conversely, PPK/GNSS provides drift-free global positioning, enabling world-aligned coordinates and ensuring long-term robustness, making it an ideal and reliable source for tasks like End2End driving. However, LiDAR odometry typically struggles in high-speed scenes or environments with few stable correspondences. Additionally, we observed that LiDAR odometry often yields inaccurate poses at the very beginning and end of a scene, while performing well in the central portion. Since object detection labels are not used for these boundary regions, small inaccuracies there are acceptable. For this reasons, we introduce a verification and selection pipeline to automatically determine which localization source is reliable for each scene, by comparing all available ego-motion estimates under a unified reference frame. Each method provides a time-ordered trajectory of poses $T_i = [R_i \,|\, t_i] \in \mathrm{SE}(3)$, where R_i and t_i denote rotation and translation, respectively. For each scene, we choose as a reference either the PPK trajectory (if available) or the GNSS coordinates otherwise, and rigidly align all LiDAR-based odometries to this reference. The alignment between a candidate trajectory and the reference is computed by estimating a global rotation R_* and translation t_* that minimize the Euclidean position discrepancy, that is

$$(R_*, t_*) \;=\; \arg\min_{R\in\mathrm{SO}(3),\, t\in\mathbb{R}^3} \sum_i \left\| R\,p_i + t - q_i \right\|_2^2, \tag{2}$$

where p_i and q_i denote corresponding positions from the candidate and reference trajectories. This least-squares problem is solved in closed form using a Kabsch alignment inside a RANSAC loop to ensure robustness against transient outliers. After alignment, we evaluate the per-frame spatial discrepancy

$$d_i = \|R_* p_i + t_* - q_i\|_2 \tag{3}$$

and measure how many points exceed a 2 m deviation threshold. A LiDAR odometry trajectory is accepted as *reliable* if fewer than ten such outliers are found across the sequence.



(a) Dataset 3D Class Labels Appearance

(b) Dataset 2D Class Labels Appearance

Figure 7. **Histogram of Occurrence of Labels in 3D and 2D in TruckDrive.** We present the full count of fine-grained instance appearance in our dataset for both 3D (left) and 2D (right), motivating the instance grouping for balanced classes representation while maintaining the high level description for all classes.

Odometry Scenes Composition. Across our dataset, the verification process identifies 1,118 scenes where PIN-SLAM provides localization of sufficient quality, and 164 scenes where MAD-ICP does so instead of PIN-SLAM. For scenes where PPK is unavailable, we use LiDAR-odometry when reliable to mitigate GNSS.

4. Depth Ground Truth

To obtain reliable depth maps for evaluating both static and dynamic parts of the scene, we construct a high-quality ground truth by combining LiDAR-based mapping and motion segmentation. Static geometry is recovered through globally consistent LiDAR accumulation, while dynamic objects are identified via radial-velocity analysis and reinserted per frame. The resulting depth maps provide temporally consistent, globally aligned, and densely annotated depth for all camera views. The process is depicted in Figure 9 and thoroughly described in this section.

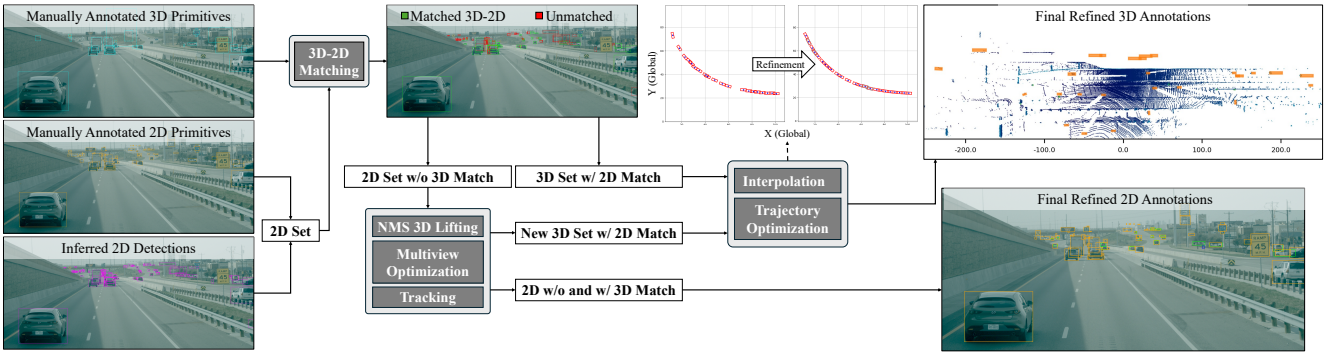


Figure 8. **Labeling Refinement.** Our labeling refinement takes as input the manually annotated 3D and 2D bounding boxes and the inferred 2D boxes. After matching 3D and 2D cuboid, we extract potential 3D candidates from un-matched 2D boxes through 3D lifting, 3D Non Maximum Suppression (NMS), multi-view optimization and tracking. The new generated 3D boxes are added to the manual annotated 3D boxes and further refined through temporal interpolation and trajectory optimization.

4.1. LiDAR Moving Point Segmentation

We derive a proxy ground truth for moving-points segmentation using radial-velocity measurements from the AEVA long-range FMCW LiDARs (which we showcase in the attached video). For each scene, we compute the distribution of positive and negative velocities separately and estimate their means (μ^+, μ^-) and standard deviations (σ^+, σ^-). A point is labeled as moving if its radial velocity exceeds $\mu^+ + 2.0\sigma^+$ (for positive velocities) or $|\mu^-| + 2.0\sigma^-$ (for negative velocities). To remove isolated outliers, we enforce spatial coherence via a KD-tree: a point retains its moving label only if at least two neighboring points within a fixed radius are also classified as moving. Finally, we retain only scenes without predominantly perpendicular motion relative to the ego vehicle, avoiding Doppler zeroes caused by tangential trajectories.

4.2. LiDAR Accumulation

Using the AEVA long-range LiDAR captures and moving-point masks from Section 4.1, we keep only static returns and place them into a single, globally aligned map. Let the ego pose at time t_i be $T_i = [R_i \,|\, t_i]$ and let p_{ik}^{ℓ} be a LiDAR return in the sensor frame. We map points to the world with the standard rigid transform

$$p_{ik}^{w} = R_i \, p_{ik}^{\ell} + t_i. \tag{4}$$

Directly concatenating all transformed points would oversample places the vehicle observes multiple times and undersample distant geometry. To obtain a balanced map with a more constant point density throughout the scene, we apply a simple range-adaptive, trajectory-aware decimation before accumulation. Around each pose we split space into distance bands $[0.0, 25.0, 60.0, 100.0, 150.0, 220.0, \infty]$. Each band uses a voxel grid and a target density: near ranges use larger voxels and lower target density, while far ranges use finer voxels and slightly higher density. Within a band, all world points falling into the same voxel compete for a small, fixed budget of samples that is proportional to the voxel volume. If a voxel lies along the planned path multiple times, we share this budget across traversals: past passes "spend" part of the budget; the remaining budget is split across the remaining passes so that, over the whole scene, each voxel ends up with roughly the intended number of points. After decimation, all selected points are transformed to the world frame and combined into a single global map:

$$\mathcal{M} = \left\{ p_{ik}^{w} \;\middle|\; k \in \mathcal{S}_i,\ i = 1, \ldots, N \right\}, \tag{5}$$

where $\mathcal{S}_i$ is the set of points retained from scan i. Finally, we apply a lightweight denoising step to remove floaters and isolated points without changing resolution: a statistical outlier filter (based on mean k-NN distance) followed by a radius-based filter (requiring at least $m = 3$ neighbors within a radius of 0.3m). The result is a globally referenced, static LiDAR map with approximately uniform, range-adaptive density and minimal artifacts from repeated traversals, that we show qualitatively in Figure 10.

4.3. Per-Frame Depth Refinement

Starting from the accumulated static point cloud in world coordinates, we generate a per-camera sparse depth map by projecting each 3D point into the image. LiDAR scans usually contain a high density of points at close range with very few

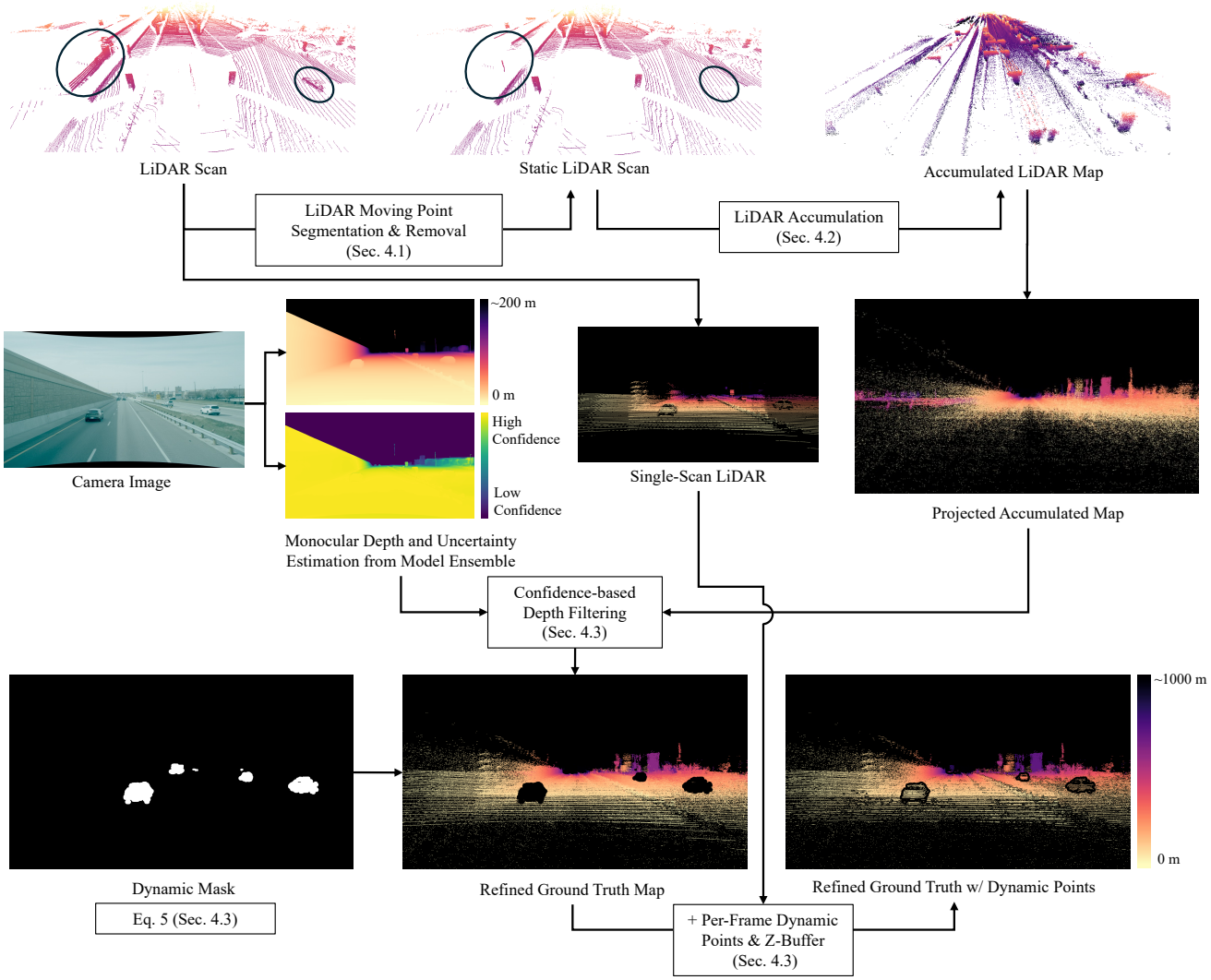


Figure 9. **Depth Ground Truth.** We present our process to generate an evaluation ground truth for the depth-estimation task. First, we remove dynamic points from the single scans and we accumulate a static LiDAR map. Second, we infer monocular depth and uncertainty from a ensemble of models for the specific view considered. Third, we refine a ground truth map joining information from the projected accumulated static map and the estimated monocular depth and uncertainty. Finally, we re-introduce dynamic objects adding the original single scan for the current frame considered.

points at large distance. The accumulated projected map yields a more balanced depth distribution, which reduces the natural bias toward near distances while more than doubling the effective maximum range. We then filter these projected depths using predictions from an ensemble of monocular depth foundation models [4, 51], removing occluded pixels whose projected depths strongly deviate from the ensemble estimate. A confidence map $C(p)$ is derived from the cross-model variance: regions where the models agree correspond to high confidence, while areas with large disagreement yield low confidence. Filtering is applied only in high-confidence regions, since monocular models are not metrically consistent but remain reliable for identifying occlusions. Given the accumulated depth $D_{\text{acc}}(p)$ and the monocular ensemble depth $D_{\text{mono}}(p)$, the filtered depth is computed as

$$D_{\text{out}}(p) = M(p)\, D_{\text{acc}}(p),$$

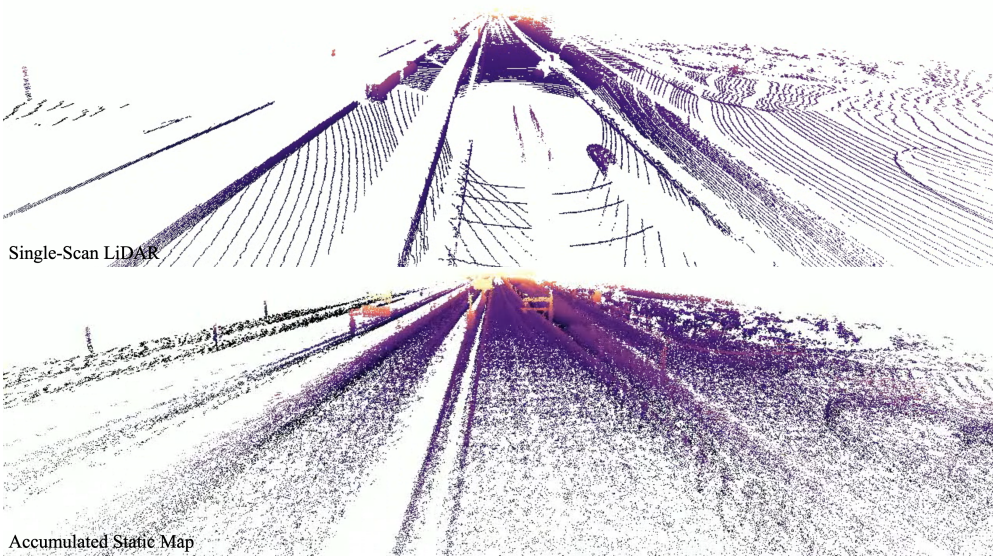


Figure 10. **LiDAR Accumulation.** We show the resulting accumulated static LiDAR map compared to a single scan. The accumulated map is characterized by a high density of 3D points, useful to generate supervision for diverse tasks such as depth estimation or occupancy forecasting.

where the binary mask $M(p)$ is defined as

$$M(p) = \begin{cases} 0, & \dfrac{|D_{\text{acc}}(p) - D_{\text{mono}}(p)|}{D_{\text{acc}}(p) + \varepsilon} > \tau_{\text{rel}} \ \wedge \ \dfrac{C(p)}{D_{\text{mono}}(p) + \varepsilon} \geq \tau_{\text{conf}}, \\ 1, & \text{otherwise.} \end{cases}$$

High confidence indicates trust in the monocular estimate, such that a large relative depth deviation justifies removing the accumulated point. We set $\varepsilon = 10^{-12}$, $\tau_{\text{rel}} = 0.5$, and $\tau_{\text{conf}} = 0.25$. Because monocular depth is typically reliable only up to approximately 200 m, while the accumulated depth can extend to nearly 1000 m, regions of low confidence bypass this filtering. In those areas, occlusions are removed using a z-buffer procedure (described in *Depth Composition*) applied as a final step. Monocular ensemble filtering remains valuable because it provides sharp depth discontinuities and preserves fine structures, complementing the z-buffer approach. This strategy follows prior work [5, 44] and yields a depth image for each camera containing only non-occluded static geometry. Unlike previous approaches, we explicitly identify dynamic objects using LiDAR-derived velocity information and incorporate them into the ground truth on a per-frame basis, as detailed below.

Dynamic Mask. Independently, we project the per-frame moving LiDAR returns (Sec. 4.1) to obtain an initial sparse binary mask M_0 of dynamic pixels. To compensate for LiDAR sparsity and small misalignments, we expand the dynamic regions using a *depth-aware* morphological dilation. Let M_1 denote the cleaned dynamic mask and D the corresponding depth map. We apply

$$M_{\text{dyn}} \ = \ \text{dilate}\big(M_1, \, r(D)\big), \tag{6}$$

where $r(D)$ specifies the dilation radius as a function of depth. In practice, we use $r(z) = 30$ px for depths below 20 m, $r(z) = 25$ px for 20 to 150 m, $r(z) = 20$ px for 150 to 200 m, and $r(z) = 10$ px beyond 200 m. This adaptive expansion ensures that nearby objects receive broader masks, filling gaps in the sparse projected points and creating dense dynamic masks.

Depth Composition. The final per-frame depth map combines static and dynamic geometry. We first remove static depth values in regions marked by the dynamic mask, ensuring that outdated accumulated background points do not overwrite moving objects. We then add back the corresponding per-frame dynamic LiDAR returns in these areas. This yields dense and temporally consistent depth maps that include both static background and accurately localized dynamic objects. Lastly, we apply a z-buffer to remove occluded points not removed by the monocular depth filter. We estimate a per-pixel front depth $F(x)$ as the nearest valid depth across all layers. A coarse scene-depth field $D_s(x)$ is then obtained by spatially smoothing

Table 3. **2D Object Detection Ranges.** We define the following range bins for the evaluation of our 2D Object Detection baselines.

Name	Meters Distance Range	Pixels Dimension Range
Short Range	0 m - 50 m	280 px - inf px
Medium Range	50 m - 150 m	93 px - 280 px
Long Range	150 m - 250 m	56 px - 93 px
Ultra Range	250 m - inf m	0 px - 56 px
Full Range	0 m - inf m	0 px - inf px

$F(x)$, which provides a global notion of relative scene distance. According to the magnitude of $D_s(x)$, we partition the image into qualitatively *near*, *mid*, and *far* depth ranges. Each range is associated with a different spatial scale, so that nearer regions use larger neighborhoods and farther regions use progressively smaller ones. In practice, these neighborhoods are defined by region-dependent radii obtained from a global base radius of 10, with near, mid, and far regions using radii proportional to $2r_{\text{base}}$, r_{base}, and $r_{\text{base}}/2$, respectively. Using these region-dependent scales, we compute an adaptive front depth $F_{\text{adapt}}(x)$ as a local minimum of $F(x)$, yielding a spatially varying estimate of the closest plausible surface around each pixel. A per-pixel occlusion ratio $\tau(x)$ is defined,tighter for near regions and looser for far regions, and a depth value $D_i(x)$ in layer i is marked as occluded whenever

$$D_i(x) > \tau(x)\, F_{\text{adapt}}(x). \tag{7}$$

The per-pixel occlusion ratio $\tau(x)$ is derived from a global base value $\tau_{\text{base}} = 1.2$ by applying region-dependent scaling: near pixels use $\tau(x) = \max(1.02,\, 0.9\,\tau_{\text{base}})$, mid-range pixels use $\tau(x) = \tau_{\text{base}}$, and far pixels use $\tau(x) = 1.1\,\tau_{\text{base}}$. Depths that do not satisfy this inequality are retained.

The resulting M_{dyn} is also retained per frame as a dynamic mask, used to distinguish foreground and background regions. For evaluation of stereo methods, we use this to report metrics separately on foreground (pixels where $M_{\text{dyn}} = 1$) and background (pixels where $M_{\text{dyn}} = 0$ and $D > 0$).

5. Baselines Details

In this section, we provide additional background together with implementation and evaluation details for all baselines used in the main paper. We also expand upon the discussion in Sec. 5 of the main manuscript, offering a more in-depth analysis of the results. We note that in all our experiments we as at most 8 V100 GPUs with $32GB$ of memory.

5.1. 2D Object Detection

We compare four 2D object detection baselines on our dataset: DETR [7], ViTDet [27], DINO [54] and YOLO11 [23]. For DETR, ViTDet and DINO, we adapt the respective MMDetection [9] implementations to our dataset. We fine tune each of them from the provided pretrained weights, training for a fixed number of epochs over the full training set. Training can be interrupted via an early stopping hook which monitors bounding box mAP with a minimum delta of 0.001 over 2 epochs. These models, as well as YOLO11 are pretrained on CoCo [29], which has 80 classes for object detection, thus we modify them to learn our dataset's 9 object classes. All results are obtained from the forward left narrow camera.

Metrics. We follow CoCo [29] and report mean average precision (mAP) averaged over 10 IoU thresholds from 0.50 to 0.95 in steps of 0.05, as well as specifically at 0.50 IoU (PASCAL VOC metric), at 0.75 IoU (strict metric), alongside mAP at short ($0 - 50$ m, SR), medium ($50 - 150$ m, MR), long ($150 - 250$ m, LR), and ultra-long-range (250+, UR). To define the range bins, we consider the average height of a car, calculate the pixel dimensions corresponding the the different ranges and filter ground truth and prediction boxes with the defined pixel dimension ranges. Given the average car height $H \approx 1.9$ m (Tab. 2) and the focal length of the considered camera (forward left narrow) $f = 7,350$ px, we calculate the pixel dimension d for a specific distance Z as

$$d = \frac{fH}{Z} = \frac{7,350\,\text{px} \cdot 1.9\,m}{Z} = \frac{13,965\,\text{px}\,m}{Z} \tag{8}$$

and define our range bins as in Table 3.

DETR [7]. DETR is a 2D object detection method which uses a ResNet-50 [19] backbone for encoding image features as input to a transformer encoder-decoder architecture, both of which have 6 layers. A key feature of DETR is its use of learned

object queries as input to the decoder to improve its performance. Fine-tuning was done as described above over 3 epochs. During training, DETR randomly selects from 11 linearly spaced input sizes ranging from 1333×480 to 1333×800, while validation is always done on 1333×800 images.

ViTDet [27]. ViTDet adopts a standard non-hierarchical ViT backbone, pretrained as a masked autoencoder and incorporated into an R-CNN–style [15] object-detection pipeline. The ViT consists of 12 layers and processes images resized to 1024×1024, which are tokenized into embeddings of dimension 768. The features output by the ViT backbone are fed into a feature pyramid network (FPN) [30] to build a *feature pyramid*, in this case at 5 different scales. Those features then fed into a region proposal network to propose candidate bounding boxes, the set of which a *region of interest* head refines to output the final boxes and classifications. Fine-tuning was done as described above over 3 epochs.

DINO [54]. DINO's network improves on DETR's object detection performance with three main additions. Firstly by introducing a denoising task alongside a contrastive objective. Secondly, by using both the encoder's outputted spatial feature maps and learnable object queries (as in DETR), as input to the transformer's decoder. Finally, a "look forward twice" scheme for bounding box prediction, wherein box predictions are refined twice per layer, rather than once. This implementation's image input shape is randomly sampled the same way as described above in DETR. It uses a Swin Transformer [31] backbone to produce 5 scales of hierarchical feature maps, which are fed into a standard DINO encoder, followed by a decoder using 900 object queries, to output bounding box and class predictions. Fine-tuning was done as described above over 3 epochs.

YOLO [23]. We use the Ultralytics library's YOLO11, specifically the 56.9 M parameter YOLO11x. The model takes an image input of shape 640×640, with padding to compensate for different aspect ratios. The images are first downsampled with a convolutional layer block, followed by a *spatial pyramid pooling - fast* block [25], *cross stage partial with spatial attention* (C2PSA), multiple *cross stage partial with kernel size* 2 (C3k2) blocks for feature fusion across scales, and finally a sequence of 2D convolutional layers, whose feature maps are passed through a set of 1×1 convolutions to predict bounding boxes, class predictions, and *objectness* scores. Fine-tuning was done as described above over 3 epochs.

Discussion. DETR, DINO,ViTDet, and the YOLO family of models are all landmark 2D object detection algorithms [34], motivating their selection as baseline methods for performance evaluation on our dataset. The results presented in Table 3, of the main manuscript, demonstrate a downward trend as the distance of the object from the camera increases. This is to be expected as these models are not adapted to high-resolution images as present in our dataset, and thus may compress distant objects to a point where they can no longer be recognized. This is especially evident in the mAP_{LR} and mAP_{UR} data, where a sharp performance drop is observed across all models. The data indicate that this has a greater influence on performance than whether the architecture employs a single-stage (DETR, YOLO11) or two-stage (DINO, ViTDet) detection scheme. For contexts such as highway driving where fine details are relevant to object detection in high-resolution images, it is therefore essential to develop new methods that can achieve this efficiently in the future. While YOLO11 is the most recent baseline studied, it is outperformed by DINO, which uses a transformer-based architecture rather than a convolutional one. This also aligns with the literature, with the tradeoff that convolutional networks are more lightweight in general and thus can be well-suited to contexts under resource constraints. We show qualitative results for the analyzed baselines in Figure 11.

5.2. 3D Object Detection

Metrics For the 3D Object Detection task we report standard Mean Average Precision (mAP) and binned mAP, evaluating predicted vs ground truth bounding boxes in the binned ranges: Short (SR $0 : 50$ m), Medium (MR $50 : 150$ m) and Long (LR $150 : 250$ m).

Far3D [22]. For our camera-only 3D object detection baseline we build on the Far3D [22] implementation in MMDetection3D, configured to operate on a longer ROI compared to the original ArgoverseV2 [47] ROI (approximately $x \in [-100, 250]$ m and $y \in [-100, 100]$ m). The model takes as input 5 surround-view images, which are resized to 2048×1550 resolution (as per original implementation), and uses the base published model configuration. We increase the depth-network maximum predicted depth from the original 100m to 250m, maintaining the number of depth bin to 50 to avoid increase in memory usage. The network is trained with AdamW [33] (learning rate $4 \cdot 10^{-4}$, weight decay 0.01) for 6 epochs.

Transfusion-L [1]. For our LiDAR-only 3D object detection baseline we build on the BEVFusion LiDAR branch [32] implementation in MMDetection3D that follows the LiDAR branch in Transfusion-L [1]. We extend the original configuration ROI to approximately $x \in [-150, 250]$ m, $y \in [-100, 100]$ m and $z \in [-16, 16]$ m to ensure a correct voxelization in the model encoding stage. We increase the voxel size from $[0.075, 0.075, 0.2]$ m to $[0.15, 0.15, 0.84$ m and conseguentely the resulting sparse shape from $[1440, 1440, 41]$ to $[2880, 1440, 41]$. The model takes as input all 7 long-range LiDARs scans and all surround short-range LiDARs, joined and transformed to the same frame of reference. The point clouds used contains: x, y, z, radial velocity, reflectivity and time-offset to ensure correct fusion among scans coming from multiple LiDARs. The

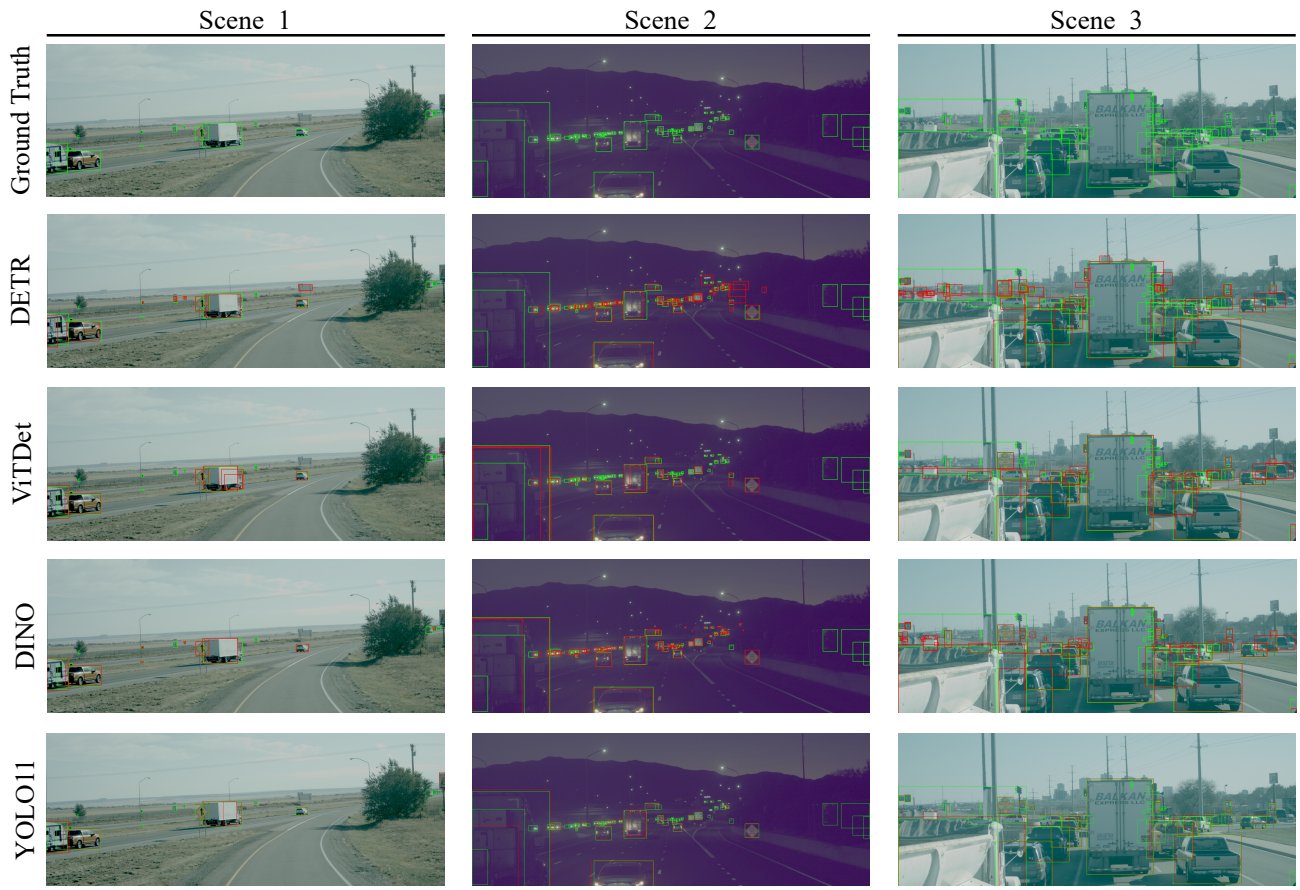


Figure 11. **Qualitative Results for 2D Object Detection.** We show qualitative comparison of 2D Object Detection baselines DETR [7], ViTDet [27], DINO [54] and YOLO11 [23]. We select three scenes with increasing difficulties (easy, night time capture and cluttered environment) and observe DINO performing the best across the tested model, especially at longer ranges.

model uses the base published model configuration. The network is trained with AdamW (learning rate $1 \cdot 10^{-4}$, weight decay 0.01) for 20 epochs with Cosine Annealing learning rate.

BEVFusion [32]. For our LiDAR-camera fusion 3D object detection baseline we build on the BEVFusion [32] implementation in MMDetection3D. We extend the original configuration ROI to approximately $x \in [-150, 250]$ m, $y \in [-100, 100]$ m and $z \in [-16, 16]$ m to ensure a correct voxelization in the model encoding stage. We increase the voxel size from $[0.075, 0.075, 0.2]$ m to $[0.15, 0.15, 0.84$ m and consequently the resulting sparse shape from $[1440, 1440, 41]$ to $[2880, 1440, 41]$. The model takes as input the same LiDAR scans as our LiDAR-only baseline and the same 5 surround view cameras as our camera-only baseline. Camera images are down sampled to the original BEVFusion resolution of 704×256. The model uses the base published model configuration, but we increase the depth-network lift-splat-shoot view transform maximum predicted depth from the original 60m to 300m and consequently the depth bins width from 0.5 m to 2.5 m to reduce the increase in memory footprint. The network is initialized from the LiDAR-only checkpoint and trained with AdamW (learning rate $1 \cdot 10^{-4}$, weight decay 0.01) for 6 epochs with Cosine Annealing learning rate.

Discussion. We choose Far3D as the best camera-only object detector at the time of the submission, while we choose BEVFusion as one of the most famous baseline for LiDAR-camera fusion model. We observe poor performances for the camera-only architecture (Far3D) for two main reasons. First, it relies on a discrete depth estimation network similar to lift-splat-shot: due to memory footprint reason, we must use relatively coarse dimensions for the depth bins, which limit accuracy for the 250 m setting. Second, there is a strong uncorrelation between objects in 2D and boxes in 3D space: in the images, objects remain visible up to roughly 1km, while in 3D we only supervise objects within 250 m, creating a sub-optimal and weak training signal for the detector. Equally, we observe limited performances of the LiDAR-only and the LiDAR-camera

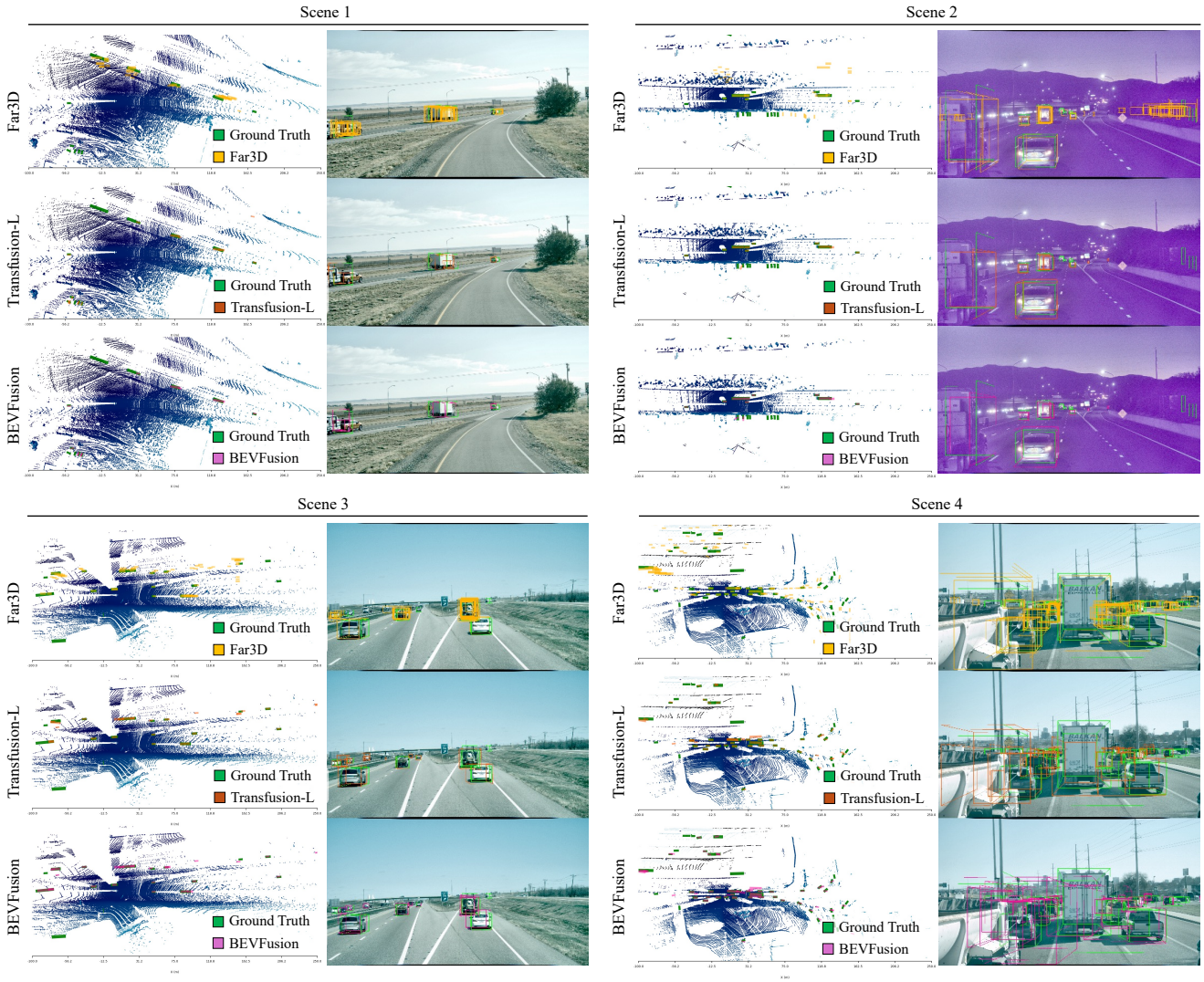


Figure 12. **Qualitative Results for 3D Object Detection.** We show qualitative comparison of 3D Object Detection baselines. Far3D [22] struggles at estimating 3D distances in BEV space, due to poor performance of the discrete depth network on the TruckDrive long ranges, while methods using the LiDAR sensors achieve overall better performance.

models: both approaches rely on dense BEV representation and extending the maximum range forces either larger grids with fixed resolution, inducing a quadratic memory growth, or coarser cells with fixed grid dimension, degrading localization and association of both smaller objects and far-range instances. We show in Figure 12 additional qualitative comparisons of the outputs of the 3 models.

5.3. 3D Tracking

Metrics. In the main manuscript, we evaluate 3D multi-object tracking performance using the public `motmetrics` library in a class-agnostic setting, where all annotated objects are treated as a single category. Tracks are associated using a distance metric defined as $d = 1 - \mathrm{IoU}_{3D}$, where IoU_{3D} is computed between oriented 3D bounding boxes by taking the intersection-over-union of their bird's-eye-view footprints and multiplying it by the overlap ratio along the vertical (z) axis; associations are only allowed if IoU_{3D} exceeds a fixed threshold, set to 0.3. From these frame-level associations, we report AMOTA (Average Multi-Object Tracking Accuracy), AMOTP (Average Multi-Object Tracking Precision) and recall. AMOTA averages MOTA over a range of operating points, jointly penalizing false positives, missed targets and identity switches across recall levels. AMOTP measures the average localization error of successfully matched objects, where the

per-association error is given by $d = 1 - \mathrm{IoU}_{3D}$, thus reflecting the quality of 3D box alignment. Finally, recall is computed as the fraction of ground-truth objects that are successfully matched to a prediction under this distance threshold. For all methods evaluation, we filter tracks with a tracking score lower than 0.3.

MUTR3D [55]. For our camera-only 3D tracking baseline we build on the MUTR3D [55] implementation in MMDetection3D, configured to operate on the same long-range ROI as our other baselines (approximately $x \in [-100, 250]$ m and $y \in [-100, 100]$ m). The model takes as input 5 surround-view images, which are resized to 704×256 resolution, and uses a ResNet-101 backbone with a deformable transformer tracking head with 300 object queries and 6 decoder layers. The BEV representation is defined on a grid with 0.2 m resolution in the horizontal plane and a single vertical bin (voxel size $[0.2, 0.2, 8]$), and the tracker predicts 3D boxes and identities. During training we sample clips of 3 consecutive frames per scene (camera-only modality) and optimize the model using a Hungarian assignment with focal loss for classification and L1 loss for box regression. The network is trained with AdamW (learning rate $2 \cdot 10^{-4}$, weight decay 0.01) for 24 epochs using a linear warm-up followed by a step decay schedule (drops at epochs 20 and 23).

Immortal Tracker [46]. We adopt an AB3DMOT-style Kalman filter with an 11-dimensional state containing position, size, yaw and their velocities, and a 7D measurement consisting of the oriented 3D box parameters. The motion model is constant-velocity in (x, y, z) and yaw, while box dimensions are treated as static. Frame-to-frame associations are computed using the Hungarian algorithm with a cost based on 2D BEV IoU between oriented rectangles parameterized by (x, y, l, w, yaw), and we discard matches with IoU below 0.3. Unmatched detections spawn new tracks, while unmatched active tracks are moved to a pool where they continue to be propagated by the Kalman filter and can be re-linked if a future detection overlaps sufficiently; tracks are only terminated after 10 consecutive frames without an update.

CenterPoint [53]. We use the original CenterPoint centertrack MMDetection3D implementation. We feed to the tracking predicted bounding boxes from our BEVFusion implementation. The tracker outputs 3D boxes with persistent IDs and class labels inherited from the underlying annotations.

Discussion. We observe poor performances of all tested tracking method on both AMOTA and AMOTP metrics. These metrics are tightly coupled to the ability of models to predict consistent and accurate bounding boxes, thus to their recall. Specifically, low IoU values lead to worse AMOTP reduce the number of frames where correct associations are possible. As a results, an offline, non-learned tracker like Immortal Tracker can perform almost as good as a learned one like CenterPoint, provided that we use the same backbone (BEVFusion) to infer $3D$ bounding boxes. Once large translation and missing or imprecise detections are involved, both trackers fail similarly, achieving only about 13% of AMOTA. MUTR3D degrade even more since, as outlined in the main manuscript, camera-based detectors struggle to infer accurate 3D object detection and to associate them, especially at mid and long ranges.

5.4. Depth Estimation

We use a distance binned MAE depth metric as a unified evaluation across all tasks, enabling direct comparison. The metric evaluates performance within fixed depth intervals—short range (0–50 m), medium range (50–150 m), long range (150–250 m), and ultra range (250–1000 m)—allowing us to quantify how methods degrade with increasing distance, a critical aspect for long range highway driving. For each bin B, the MAE is computed only over pixels whose ground truth depth lies inside that bin:

$$\mathrm{MAE}_B = \frac{1}{|\Omega_B|} \sum_{x \in \Omega_B} \left| \hat{d}(x) - d^{\mathrm{gt}}(x) \right|, \qquad \Omega_B = \{x \mid d^{\mathrm{gt}}(x) \in B\}.$$

This unified formulation makes performance directly comparable across all tasks and benchmarks. In the following, we provide details and discussion of the implemented baselines for the three tasks, Surround Depth Estimation, Stereo Depth Estimation and Monocular Depth Estimation and describe the task specific metrics we employ alongside our unified metrics to ensure comparability with existing benchmarks. In Figure 13 we report qualitative results of the baselines analyzed in the manuscript for three scenes.

5.4.1. Surround Depth Estimation

Surround depth estimation is a central perception task in autonomous driving, enabling 3D scene understanding from camera-only sensor suites. Beyond multi-view inputs, many methods also exploit short temporal queues to improve geometric consistency over time, which our dataset supports through its densely sampled sequential recordings. However, incorporating temporal information remains challenging in dynamic scenes where object motion breaks frame-to-frame correspondence. Our dataset provides up to 15 synchronized and calibrated cameras with substantial inter-camera overlap, in sequential recordings, allowing reliable multi-view and multi-frame triangulation. For benchmarking, we implement baselines using 5 cameras that still cover most of the 360° surround while retaining sufficient overlap between neighboring cameras.

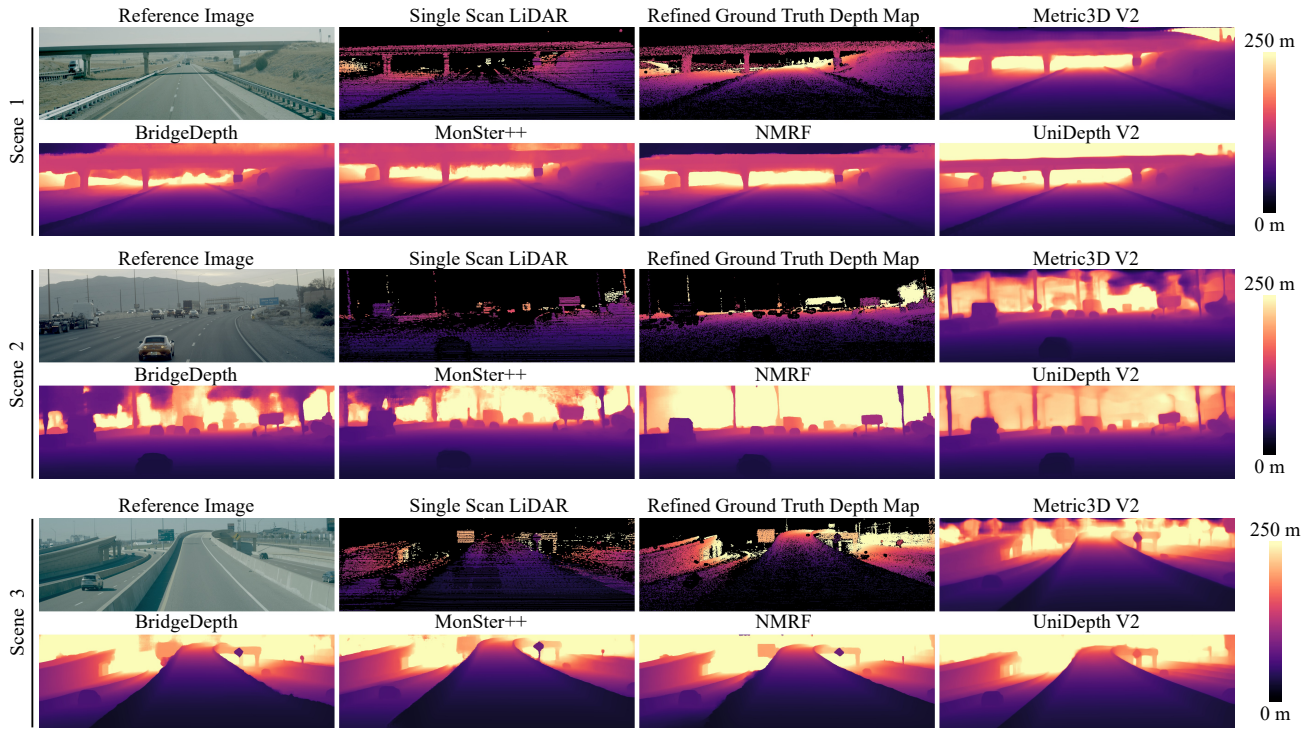


Figure 13. **Qualitative Results for Depth Estimation.** We show qualitative comparison of different depth estimation baselines together with sparse LiDAR and accumulated ground truth on the forward facing, long focal length camera. Monocular depth methods [20, 38] produce visually appealing results however, metrical accuracy is lower compared to stereo methods [11, 16, 17]. Although the refined ground-truth depth map spans depths of up to 1000 m, we restrict the colorization of the visualizations to 250 m to maintain interpretability of the depth maps.

Metrics. We follow the task-specific multi-frame depth evaluation protocol of [40]. Given the estimated depths d_t^c and ground-truth depth d_t^{*c}, we compute the following depth metrics:

$$\text{Abs Rel} = \frac{1}{T \cdot C} \sum_{d,c} \frac{|d_t^c - d_t^{*c}|}{d_t^{*c}}, \tag{9}$$

$$\text{Sqr Rel} = \frac{1}{T \cdot C} \sum_{d,c} \frac{\|d_t^c - d_t^{*c}\|^2}{d_t^{*c}}, \tag{10}$$

$$\text{RMSE} = \frac{1}{T \cdot C} \sqrt{\sum_{d,c} \|d_t^c - d_t^{*c}\|^2}, \tag{11}$$

$$\delta_{1.25} = \text{fraction of } d \in \mathcal{D} \text{ for which } \max\left(\frac{d}{d^*}, \frac{d^*}{d}\right) < 1.25. \tag{12}$$

R3D3 [40]. In its original form, R3D3 is trained in two stages, where the first stage pre-processes entire sequences using a variant of DROID-SLAM [43] to generate geometric priors. We adapt this stage to an online setting that incrementally builds the graph from the current frame and the previous 5 timestamps and use it to compute geometric priors during inference. Since the original implementation only provides pretrained weights for nuScenes [6] and DDAD [18] and does not include training code for this stage, we directly adopt these released weights. The second stage is trained on our dataset with LiDAR supervision, using a simple mean absolute error (MAE) loss between the predicted depth $\hat{d}$ and the projected LiDAR ground truth d^{gt}:

$$\mathcal{L}_{\text{MAE}} = \frac{1}{|\Omega|} \sum_{x \in \Omega} |\hat{d}(x) - d^{\text{gt}}(x)|, \tag{13}$$

where Ω denotes the set of pixels with valid LiDAR depth. We initialize this stage from the official nuScenes [6] pretrained weights and train it for 1 epoch on our training split. The input and output resolution is 1280×704 pixels.

MapAnything [24]. MapAnything is a recent method built on top of VGGT [45], capable of ingesting intrinsic and extrinsic calibrations to produce metric depth and 3D reconstructions. We start from the officially released pretrained weights and finetune the model on our dataset by providing 5 images from a single timestamp together with the corresponding camera intrinsics and extrinsics. Finetuning is performed for 1 epoch on our training split using the MAE loss with projected LiDAR supervision (Eq. 13). Since MapAnything supports up to 518 pixels on its maximum spatial dimension, we use an input and prediction resolution of 518×294.

Discussion. R3D3 [40] is an early approach that integrates SLAM-based geometric priors into monocular depth estimation, making it a useful baseline for our setting. MapAnything [24], built on VGGT [45], represents a recent class of large-scale 3D reconstruction models that operate in metric scale and are relevant for autonomous driving. In our experiments, MapAnything reaches accuracy comparable to metric monocular depth methods but is still outperformed by most dedicated approaches. Its possibility for the use of multiple frames and calibrated poses, however, points toward a promising direction for temporally informed, continuous 3D reconstruction.

5.4.2. Stereo Depth Estimation

Stereo depth estimation infers depth by predicting disparities between corresponding pixels in a rectified stereo pair. Depth is then obtained via

$$d = \frac{f \cdot b}{\delta}, \tag{14}$$

where d denotes depth, f the focal length, b the stereo baseline, and δ the predicted disparity. Recent approaches combine disparity estimation with monocular depth foundation models, injecting strong priors learned from large-scale datasets. This improves zero-shot performance and alleviates typical stereo failure modes, such as textureless or reflective regions where reliable correspondences are hard to obtain [11, 17]. Our dataset includes a horizontal stereo pair with an ultra-wide baseline of approximately 1.57 m and a focal length of about 7350 pixels, which yields more than 10 pixels of disparity even at distances of 1000 m when using the full image resolution of 3848×2168 pixels. The truck setup additionally includes two vertical stereo pairs facing rearward, forming a configuration that deviates from the classical horizontal stereo assumption. All baselines in our study are implemented using the front-facing horizontal stereo pair, predicting disparity for the front left narrow camera.

Metrics. In the automotive domain, stereo depth estimation is commonly evaluated on the KITTI Stereo benchmark [14], which reports the standard D1 error metrics. We follow this protocol and compute errors in disparity space. Let $\delta(x)$ and $\delta^*(x)$ denote the predicted and ground-truth disparities at pixel x, respectively, and define

$$\Omega_{\text{gt}} = \{x \mid \delta^*(x) > 0\} \tag{15}$$

as the set of pixels with valid ground-truth disparity. For each $x \in \Omega_{\text{gt}}$, we use

$$e_{\text{abs}}(x) = \left|\delta(x) - \delta^*(x)\right|, \qquad e_{\text{rel}}(x) = \frac{e_{\text{abs}}(x)}{\left|\delta^*(x)\right|}. \tag{16}$$

A pixel is counted as an outlier if

$$e_{\text{abs}}(x) > 3.0 \quad \text{and} \quad e_{\text{rel}}(x) > 0.05. \tag{17}$$

Foreground and background regions are defined via a binary foreground mask $M_{\text{fg}}(x)$ obtained from LiDAR, as described in Sec. 4.2:

$$\Omega_{\text{bg}} = \{x \in \Omega_{\text{gt}} \mid M_{\text{fg}}(x) = 0\}, \tag{18}$$

$$\Omega_{\text{fg}} = \{x \in \Omega_{\text{gt}} \mid M_{\text{fg}}(x) \neq 0\}, \tag{19}$$

$$\Omega_{\text{all}} = \Omega_{\text{gt}}. \tag{20}$$

Let $\Omega_{\text{bg}}^{\text{out}}$, $\Omega_{\text{fg}}^{\text{out}}$, and $\Omega_{\text{all}}^{\text{out}}$ denote the subsets of pixels in Ω_{bg}, Ω_{fg}, and Ω_{all} that satisfy the outlier condition above. The D1 error rates are then

$$\text{D1-bg} = 100 \cdot \frac{|\Omega_{\text{bg}}^{\text{out}}|}{|\Omega_{\text{bg}}|}, \quad \text{D1-fg} = 100 \cdot \frac{|\Omega_{\text{fg}}^{\text{out}}|}{|\Omega_{\text{fg}}|}, \quad \text{D1-all} = 100 \cdot \frac{|\Omega_{\text{all}}^{\text{out}}|}{|\Omega_{\text{all}}|}. \tag{21}$$

Across our experiments, we restrict evaluation to dense stereo methods, such that prediction density is effectively 100% for all of them and is therefore not explicitly repeated in every table.

NMRF [16]. NMRF is a stereo estimation method that reformulates classical MRF-based stereo matching using neural networks. It replaces hand-crafted potentials and message passing with learned neural components and introduces a Disparity Proposal Network to adaptively restrict the disparity search space. We finetune NMRF on $10\,\mathrm{k}$ samples from our training split using the official training code and hyperparameters, starting from the authors' KITTI-pretrained checkpoint. To fit the model within GPU memory limits, we reduce the input resolution by roughly one third to 1280×720 and apply a vertical crop in the sky region, yielding a final training resolution of 1280×432 and train with batch size 1.

MonSter++ [11]. MonSter++ is a stereo depth estimation method that augments correspondence-based matching with monocular depth priors. Its dual-branch architecture fuses monocular and stereo cues, allowing each to guide the other in regions with weak or ambiguous correspondences. We finetune MonSter++ from the authors' KITTI-pretrained checkpoints using the official training code and hyperparameters, employing the same input resolution and number of training steps as used for NMRF [16].

BridgeDepth [17]. BridgeDepth extends NMRF by jointly leveraging monocular and stereo cues for depth estimation. Its core cross-attentive alignment module bidirectionally fuses monocular structure priors with stereo matching features to improve performance in ambiguous regions. We finetune BridgeDepth from the authors' KITTI-pretrained checkpoints using the official training code and hyperparameters, employing the same input resolution and number of training steps as used for NMRF [16].

Discussion. All stereo baselines—BridgeDepth [17], MonSter++ [11], and NMRF [16]—were chosen as recently published methods that report competitive performance on standard benchmarks such as KITTI [14]. BridgeDepth and MonSter++ fuse monocular and stereo predictions within their pipelines, while NMRF follows a more classical stereo-only formulation. In our evaluations, all stereo approaches outperform monocular methods, especially at longer ranges, highlighting the importance of triangulation and geometric constraints for achieving the accuracy required in long-range highway scenes. BridgeDepth, which extends NMRF, shows improved performance at short ranges but underperforms at longer distances, suggesting that the integrated monocular component aids near-field depth estimation but introduces priors that become less reliable as distance increases.

5.4.3. Monocular Depth Estimation

Monocular depth estimation aims to infer scene geometry from a single image. However, this problem is inherently ill-posed, as a single view can correspond to infinitely many plausible 3D interpretations. Recently, large-scale depth foundation models have emerged that generalize across diverse datasets, but many of these models focus on predicting relative depth, making them practical for tasks such as segmentation, normal estimation, or scene understanding, but not directly usable for metric 3D reconstruction. To address this limitation, several approaches incorporate camera parameters—most notably the focal length—into the prediction process and leverage large-scale curated training data, enabling monocular depth networks to produce metric depth estimates suitable for downstream geometric applications such as required in autonomous driving.

Metrics. Monocular depth estimation in the automotive domain is commonly evaluated on the KITTI Depth Estimation benchmark [44]. In our work, we follow this protocol and report the four standard metrics SiLog, sqERel, absERel, and iRMSE. Let $d(x)$ and $d^*(x)$ denote the predicted and ground-truth depth at pixel x, and define the set of valid ground-truth pixels as

$$\Omega = \{x \mid d^*(x) > 0\}. \tag{22}$$

All metrics below are averaged over $x \in \Omega$. Let

$$e_{\log}(x) = \log d(x) - \log d^*(x). \tag{23}$$

The scale-invariant log error (SiLog) is

$$\mathrm{SiLog} = \sqrt{\frac{1}{|\Omega|}\sum_{x\in\Omega} e_{\log}(x)^2 - \left(\frac{1}{|\Omega|}\sum_{x\in\Omega} e_{\log}(x)\right)^2}. \tag{24}$$

$$\mathrm{sqERel} = \frac{1}{|\Omega|}\sum_{x\in\Omega} \frac{\big(d(x) - d^*(x)\big)^2}{\big(d^*(x)\big)^2} \tag{25}$$

$$\text{absERel} = \frac{1}{|\Omega|} \sum_{x \in \Omega} \frac{\left| d(x) - d^*(x) \right|}{d^*(x)} \tag{26}$$

$$\text{iRMSE} = \sqrt{\frac{1}{|\Omega|} \sum_{x \in \Omega} \left(\frac{1}{d(x)} - \frac{1}{d^*(x)} \right)^2} \tag{27}$$

These definitions correspond directly to the metrics computed by the official KITTI depth evaluation code. Notably, the definition of sqERel differs from the Sqr Rel metric used for surround depth estimation (Sec. 5.4.1), as the denominator uses the squared ground-truth depth.

ZoeDepth [2]. ZoeDepth is a monocular depth estimation method designed to achieve both strong generalization and metric depth prediction. It accomplishes this by combining large-scale relative-depth pretraining with a metric fine-tuning stage that uses a dedicated "metric bins" module to recover absolute scale. The model uses the DPT-BEiT-Large [39] backbone at 512×384 input resolution. We train the full ZoeDepth network, initializing with the official pre-trained weights, with the proposed parametrization for 1 epoch, using a one-cycle style schedule and a reduced learning rate ($\times 0.1$) for the MiDaS [3] backbone while keeping the encoder and positional encodings at the base rate. Training uses gradient clipping at 0.1 and mild data augmentation with small random in-plane rotations ($\pm 1^\circ$) and no random cropping. We train for one epoch and evaluate predictions in the range $[0, 400]$ m without Eigen/Garg cropping.

Metric3Dv2 [20]. Metric3Dv2 is a monocular depth estimation method designed to recover metric depth rather than the affine-invariant depth typical of foundation models. It resolves the metric ambiguity across diverse camera models through a canonical camera-space transformation, enabling zero-shot metric depth prediction from a single image. We train Metric3Dv2 for one epoch on our training split using an MAE loss (Eq. 13) computed from projected LiDAR depth. We use the DINO2reg-ViT-Small [8] configuration, initialize with the official pre-trained weights and optimize with AdamW using a learning rate of $1\text{e}{-5}$ and a batch size of 2.

UniDepthV2 [38]. UniDepthV2 achieves metric monocular depth by directly predicting metric 3D points from a single input image, removing the need for any additional camera information at inference time. A self-promptable camera module estimates a dense camera representation that conditions the depth features, enabling consistent metric reconstruction across domains. We use the ViT-L 14 backbone with the officially released pretrained weights and finetune the model for one epoch on our training split using an MAE loss (Eq. 13) with an AdamW optimizer and a learning rate of $1\text{e}{-6}$. Input images are downsampled to a resolution of 1274×434 to satisfy the network's requirement of having spatial dimensions divisible by 32.

Discussion. All depth methods are finetuned twice, once predicting depth for the same camera as the stereo methods and once predicting depth for the 5 cameras used in the surround depth estimation task, making direct comparison possible. Across our benchmarks, monocular methods outperform current multiview approaches, but while they remain competitive with stereo at short ranges, they fall behind at medium and long distances, where explicit triangulation provides stronger geometric cues.

5.5. Temporal Scene Modeling and Reconstruction

5.5.1. LiDAR Forecasting

Metrics. For the LiDAR forecasting tasks we allign to prior work [52] and computer Chamfer Distance (CD) between predicted and ground truth point clouds. 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

$$CD = \frac{1}{2N} \sum_{\mathbf{x} \in \mathbf{X}} \min_{\hat{\mathbf{x}} \in \hat{\mathbf{X}}} \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 + \frac{1}{2M} \sum_{\hat{\mathbf{x}} \in \hat{\mathbf{X}}} \min_{\mathbf{x} \in \mathbf{X}} \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 \tag{28}$$

where $\mathbf{x} \in \mathbf{R}^{Nx3}, \hat{\mathbf{x}} \in \mathbf{R}^{Mx3}$ represent the ground-truth and predicted point clouds, respectively. We additionally report L1 error between fore-casted depths along each LiDAR ray $\hat{\mathbf{D}} \in \mathbf{R}^N$ and ground truth depths $\mathbf{D} \in \mathbf{R}^M$ as

$$L1 = mean(abs(\mathbf{D} - \hat{\mathbf{D}})) \tag{29}$$

4DOcc [26]. For 4DOcc results we rely on the code base that was provided with the the publication. The model is extended to the ROI of $[-100, 100]$ m on y axis, $[-100, 250]$ m 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 [52]. For ViDAR results we build on top of the publicly available implementation by the original authors. The model is extended to the ROI of $[-100, 100]$ m on y axis, $[-100, 250]$ m 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 3 s 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 TruckDrive Dataset.

LRS4Fusion [36]. For LRS4Fusion, we implement the model as detailed in their manuscript and train it with the same ROI of $[-100, 100]$ m on y axis, $[-100, 250]$ m on x and voxelize the input point cloud with voxels of dimension 0.20 m in the x and y axis and 0.38 on the z axis. The model is trained with the reported scheduling. We train LRS4Fusion using the same five cameras employed by the baselines for the 3D object detection task.

Discussion. We chose 4DOcc, ViDAR and LRS4Fusion as they are competitively performing models and represent a strong camera-only, LiDAR-only and LiDAR-camera forecasting baselines. 4DOcc struggles to extract temporal information from the history horizon due to the large motion between frame in the highway scenarios. ViDAR is less affected by large scene movements due to its use of expensive deformable attention and multi-frame reasoning but still struggles with accurate long-range 3D geometry, as monocular surround cameras lack the necessary depth cues and geometric constraints for estimating distant structures. LRS4Fusion is specifically designed for efficiency and performance on long-range scenarios and therefore performs best among the baselines.

5.5.2. Moving Object Segmentation

Metrics. We evaluate LiDAR moving object segmentation with two standard region-based metrics: Intersection over Union (IoU) and F1 score. IoU measures the overlap between the predicted moving mask and the ground-truth moving points, penalizing both over and under segmentation. The F1 score is the harmonic mean of precision and recall, capturing the balance between correctly detected moving points and false alarms or misses. We enable training and evaluation on per-point binary labels (*moving* vs. *static*) extracted as described in Section 4.1.

4DMOS [35]. For 4DMOS we rely on the official released code base. We train the model on our data with a voxel size of 0.1 m, with a scan period of $\Delta t = 0.1$ s and a queue of $N_{\text{past}} = 10$ past LiDAR frames, so that the network sees a temporal window of 1 s history at the native sensor rate. A Gaussian weighting scaled by a factor of 2 is applied to our moving objects ground truth and the original training schedule is preserved. At inference time, we further apply the binary Bayes filter strategy from the original work at 0.1 s temporal resolution.

Discussion. We select 4DMOS for its reported generalization capabilities and thus test first its pretrained, public checkpoint on TruckDrive data. We observe that the model accurately perceives moving actors, but rarely at distances higher than 50 meters. After training on our long-range data, performance improves markedly, but still falls short of the 60–70% IoU reported on other benchmarks, reaching only about 32% IoU on our dataset. This gap highlights an architectural lack in the spatio-temporal feature learning based on sparse grids, consistent with what we observed for object detection and tracking. We note additionally, that in order to train 4DMOS, which still leverages sparse representations, with the reported settings we are bounded, by our GPUs memory capacity, to use a batch size of 1, highlighting the need for more efficient architectures for long range perception.

5.5.3. Scene Reconstruction

Metrics. We evaluate all scene reconstruction baselines using standard full-reference image quality metrics: Peak signal-to-noise ratio (PSNR), which measures the pixel-wise fidelity between rendered RGB images and ground-truth frames, and structural similarity index (SSIM), that assesses perceptual similarity by comparing local luminance, contrast, and structure. Unless otherwise stated, we report PSNR and SSIM averaged over all evaluation views and timesteps for each method, ensuring a fair comparison across models and reconstruction backbones.

OmniRE [10]. We build on the original dynamic 3D Gaussian Splatting implementation, and adapt it to our highway setting by using ground-truth 3D bounding boxes to initialize per-instance Gaussians for moving objects and a LiDAR-based background (up to 8×10^5 points plus random samples) for the static scene. To reduce memory, we downscale input images to half resolution and uniformly subsample LiDAR by a factor of 4 before initialization, and use SegFormer [48] to derive sky masks that are modeled with an environment lighting module. We render from two forward-facing cameras per scene using ground-truth camera and LiDAR poses, and train for $30{,}000$ iterations with a multi-resolution schedule, combining RGB, SSIM, mask, and weak LiDAR depth losses plus standard Gaussian regularizers. For moving vehicles we enable rigid

Figure 14. **Qualitative Reconstruction Comparison.** We show qualitative comparison of reconstruction between NERF based model Dynamic Nerfacto [42] and 3D Gaussian Splatting based model OmniRE [10]. The second overall excel in dynamic scene reconstruction compared to his counterpart as well as in the overall background reconstruction. With the last example, we highlight how both suffer more from strong driving maneuvers like a turn.

instance nodes and add a lightweight vertical regularization on rigid translation to discourage spurious motion along the z axis.

HUGS [56]. We adapt the public HUGS implementation with its default training setup, and pre-process our data with semantic segmentation, a COLMAP [41] initialization, and monocular depth priors from UniDepthv2 [38]. We train for $30{,}000$ iterations and follow the original densification schedule: Gaussians are densified between iterations 500 and $15{,}000$ based on gradient magnitude, with opacity resets every $3{,}000$ steps. Optimization jointly updates Gaussian positions, scales, rotations, colors, opacities, and semantic features with separate learning rates, using an RGB+DS-SIM objective (with $\lambda_{\text{DSSIM}} = 0.2$) plus smoothness and distance regularizers on the semantic field. As in the original HUGS configuration, we also apply a ground-plane prior using a $0.2\,\text{m}$ grid in the band $[-2, 2]\,\text{m}$ to bias Gaussians toward the road surface.

DynamicNerfacto [42]. We use the *dynamic-nerfacto* model from Nerfstudio with the official trainer on our TruckDrive multi-camera sequences. We train with up to six cameras (two forward narrow, two rear medium, two side-front wide), Nerfstudio's dynamic dataloader and ground-truth 3D bounding boxes, without any depth supervision. To control memory, we downscale all camera streams by a factor of 0.3 and sample $4{,}096$ rays per batch, training for $100{,}000$ iterations with mixed precision. We keep the default average initial density (0.01) and use the standard Nerfstudio exponential learning-rate schedules for fields, proposal networks, camera optimization, trajectory optimization, hash grids, and CNN components.

Discussion. We select these three baselines to span complementary families of dynamic scene reconstruction methods: a modern NeRF-style model (Dynamic Nerfacto) and two state-of-the-art 3D Gaussian Splatting approaches (HUGS and OmniRE). Both 3DGS methods substantially outperform the NeRF baseline on our long-range, multi-camera highway setting, with HUGS improving PSNR and SSIM over Dynamic Nerfacto, and OmniRE yielding the best overall reconstruction quality. HUGS initializes and refines unicycle models to handle dynamic gaussians, while OmniRE benefits from the instance-aware initialization with ground-truth 3D boxes, LiDAR-guided background modeling and explicit handling of moving objects nodes, leading to a gain of roughly $7.5\,\text{dB}$ PSNR and 0.08 SSIM over Dynamic Nerfacto. Overall, these results suggest that, in our highway regime, Gaussian Splatting representations are more effective than NeRF-style fields and ground truth bounding boxes are fundamental to aid the model in learning large translations for dynamic objects. We compare qualitatively 3DGS with NERF methods in Figure 14.

5.6. End-to-End Planning

Metrics. For the End-to-End Planning task we follow UniAD [21] and report L2 error between the refined ground truth ego-pose and the predicted pose for the 6 future timestamps, corresponding to the next $0.2, 0.4, 0.6, 0.8, 1.0, 1.2$ seconds. We do not include the Collision Rate metric in the main manuscript as not indicative of the driving performances due to two main reasons. First, highway scenes are sparser then urban scenes, with higher minimum distances between ego and other actors, making collisions much less common. Second, the collision metric is computed from a discrete collision map generated by projecting bounding boxes in BEV, where the number of grids is defined by the occupancy network of the model; such network produces an output image of 200×200, effectively encoding $2.5 \times 2.5\,\text{m}$ grid in a single pixel, too coarse to give a

Table 4. **E2E Planning.** We train UniAD [21] on TruckDrive and evaluate L2 error and Collision Rate on the 6 future timestamps predictions, corresponding to the next $0.2, 0.4, 0.6, 0.8, 1.0, 1.2$ seconds.

Method	$L2$ **(m)**↓							$Collision$ **(%)**↓						
	1 step	**2 step**	**3 step**	**4 step**	**5 step**	**6 step**	**Avg.**	**1 step**	**2 step**	**3 step**	**4 step**	**5 step**	**6 step**	**Avg.**
UniAD [21]	0.57	1.13	1.71	2.30	2.88	3.42	2.00	1.228	1.172	1.010	0.790	0.766	0.743	0.952

Table 5. **UniAD Additional Results**. We report per-task results of UniAD. Tracking: AMOTA (Average Multi-object Tracking Accuracy), distance based AMOTP (Average Multi-object Tracking Precision) and IDS (Identity Switches). Occupancy: IoU and VPQ (Video Panoptic Quality). Planning: avg.L2 and avg.Col (L2 error and Collision Rate averaged values across the planning horizon). We note how tracking task struggles to converge and the few high confidence predicted agents are not tracked consistently across time causing ids switches.

Tracking				**Occupancy Prediction**			**Planning**	
Range [m]	**AMOTA**↑	**AMOTP**↓	**IDS**↓	**Range** $[m^2]$	**IoU**↑	**VPQ**↑	**avg.L2**↓	**avg.Col.**↓
0 to 50	0.0	3.0125	324	50×50	0.18	0.0	2.00 m	0.95 %
50 to 150	0.0	2.9266	353	100×100	0.44	0.0		
150 to 250	0.0	3.2181	85	250×250	0.34	0.0		
FULL	0.0	3.1254	878	500×500	0.34	0.0		

real informative metric. We report the full L2 and Collision Rate metric in Table 4.

UniAD [21]. For our End-to-End Planning baseline we choose UniAD as one of the most famous and a pioneer work in the field. We build on the public released MMDetection3D implementation. We keep the underling model untouched but we replace the camera-only BEV encoder with a LiDAR-only architecture based on Second [49], as the original setting was unable to converge. Similarly to the original setting, we implement past-information fusion in the form of two small BEV convolution layers after concatenating the current and the previous BEV feature maps. In absence of HD-Map data, we additional deactivate the map segmentation head. We extend the original configuration ROI to $x \in [-250, 250]$ m, $y \in [250, 250]$ m and $z \in [-12.5, 12.5]$ m to ensure a correct voxelization in the model encoding stage. We set the voxel size to $[0.9765625, 0.9765625, 25]$ m and consequently the resulting sparse shape to $[4100, 1600, 16]$. The model takes as input the same LiDAR scans as our 3D Object Detection baselines. We set the queue length to 3 to fit the training on our GPUs memory. We follow the original training regime and subdivide the learning in two stages: in Stage 1 we train the BEV Backbone, the Object Detection head and Tracking head while in Stage 2 we train the full model in an End-to-End fashion, additionally activating the Motion Forecasting, the Collision Map and the Planning heads. We initialize our weights from the pre-trained NuScenes checkpoint and train Stage 1 and Stage 2 for 4 epochs each with AdamW (learning rate $2 \cdot 10^{-4}$, weight decay 0.01, 500 iteration of warm-up).

Discussion. UniAD exhibits consistently weak performance across all components of the pipeline, see Table 5, which we primarily attribute to the extremely coarse BEV representation enforced by the 200×200 grid and to the substantial difficulty of optimizing the full architecture over our extended 250×250 m spatial range. The limited BEV resolution propagates through the detection, tracking, forecasting, and planning heads, constraining the model's ability to encode fine-grained scene geometry and actor interactions. Furthermore, despite clear signs of underfitting in the perception tasks and a planning head that degenerates toward straight-line behaviors (see supplementary videos), the final open-loop L2 metric remains low. This ambiguity reinforces observations from prior work [12, 28] that open-loop error alone fails to reflect actual closed-loop drivability, especially in sparse highway environments where small trajectory deviations do not immediately translate into safety-critical events. Overall, these results highlight the limitations of current end-to-end architectures when deployed in large-scale, long-range settings and underscore the need for higher-fidelity spatial representations and stronger optimization regimes.

References

[1] Xuyang Bai, Zeyu Hu, Xinge Zhu, Qingqiu Huang, Yilun Chen, Hongbo Fu, and Chiew-Lan Tai. Transfusion: Robust lidar-camera fusion for 3d object detection with transformers, 2022. 15

[2] Shariq Farooq Bhat, Reiner Birkl, Diana Wofk, Peter Wonka, and Matthias Müller. Zoedepth: Zero-shot transfer by combining relative and metric depth, 2023. 22

[3] Reiner Birkl, Diana Wofk, and Matthias Müller. Midas v3.1 – a model zoo for robust monocular relative depth estimation. *arXiv preprint arXiv:2307.14460*, 2023. 22

[4] Aleksei Bochkovskii, Amaël Delaunoy, Hugo Germain, Marcel Santos, Yichao Zhou, Stephan R. Richter, and Vladlen Koltun. Depth pro: Sharp monocular metric depth in less than a second. In *International Conference on Learning Representations*, 2025. 12

[5] Samuel Brucker, Stefanie Walz, Mario Bijelic, and Felix Heide. Cross-spectral gated-rgb stereo depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21654–21665, 2024. 13

[6] 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. 19, 20

[7] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 14, 16

[8] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021. 22

[9] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 14

[10] Ziyu Chen, Jiawei Yang, Jiahui Huang, Riccardo de Lutio, Janick Martinez Esturo, Boris Ivanovic, Or Litany, Zan Gojcic, Sanja Fidler, Marco Pavone, Li Song, and Yue Wang. Omnire: Omni urban scene reconstruction. In *The Thirteenth International Conference on Learning Representations*, 2025. 23, 24

[11] Junda Cheng, Wenjing Liao, Zhipeng Cai, Longliang Liu, Gangwei Xu, Xianqi Wang, Yuzhou Wang, Zikang Yuan, Yong Deng, Jinliang Zang, Yangyang Shi, Jinhui Tang, and Xin Yang. Monster++: Unified stereo matching, multi-view stereo, and real-time stereo with monodepth priors, 2025. 19, 20, 21

[12] Daniel Dauner, Marcel Hallgarten, Tianyu Li, Xinshuo Weng, Zhiyu Huang, Zetong Yang, Hongyang Li, Igor Gilitschenski, Boris Ivanovic, Marco Pavone, Andreas Geiger, and Kashyap Chitta. Navsim: Data-driven non-reactive autonomous vehicle simulation and benchmarking, 2024. 25

[13] 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. 9

[14] 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. 20, 21

[15] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Region-based convolutional networks for accurate object detection and segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(1):142–158, 2016. 15

[16] 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. 19, 21

[17] Tongfan Guan, Jiaxin Guo, Chen Wang, and Yun-Hui Liu. Bridgedepth: Bridging monocular and stereo reasoning with latent alignment. *To appear*, pages 27681–27691, 2025. 19, 20, 21

[18] Vitor Guizilini, Rares Ambrus, Sudeep Pillai, Allan Raventos, and Adrien Gaidon. 3d packing for self-supervised monocular depth estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 19

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 14

[20] Mu Hu, Wei Yin, Chi Zhang, Zhipeng Cai, Xiaoxiao Long, Hao Chen, Kaixuan Wang, Gang Yu, Chunhua Shen, and Shaojie Shen. Metric3d v2: A versatile monocular geometric foundation model for zero-shot metric depth and surface normal estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. 19, 22

[21] Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, Lewei Lu, Xiaosong Jia, Qiang Liu, Jifeng Dai, Yu Qiao, and Hongyang Li. Planning-oriented autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. 24, 25

[22] 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. 15, 17

[23] Glenn Jocher and Jing Qiu. Ultralytics yolo11, 2024. 14, 15, 16

[24] Nikhil Keetha, Norman Müller, Johannes Schönberger, Lorenzo Porzi, Yuchen Zhang, Tobias Fischer, Arno Knapitsch, Duncan Zauss, Ethan Weber, Nelson Antunes, Jonathon Luiten, Manuel Lopez-Antequera, Samuel Rota Bulò, Christian Richardt, Deva Ramanan, Sebastian Scherer, and Peter Kontschieder. MapAnything: Universal feed-forward metric 3D reconstruction. In *arXiv:2509.13414*, 2025. 20

[25] Rahima Khanam and Muhammad Hussain. Yolov11: An overview of the key architectural enhancements, 2024. 15

[26] Tarasha Khurana, Peiyun Hu, David Held, and Deva Ramanan. Point cloud forecasting as a proxy for 4d occupancy forecasting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 22

[27] Yanghao Li, Hanzi Mao, Ross Girshick, and Kaiming He. Exploring plain vision transformer backbones for object detection. *arXiv preprint arXiv:2203.16527*, 2022. 14, 15, 16

[28] Zhiqi Li, Zhiding Yu, Shiyi Lan, Jiahan Li, Jan Kautz, Tong Lu, and Jose M. Alvarez. Is ego status all you need for open-loop end-to-end autonomous driving?, 2024. 25

[29] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015. 14

[30] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 15

[31] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10012–10022, 2021. 15

[32] Zhijian Liu, Haotian Tang, Alexander Amini, Xingyu Yang, Huizi Mao, Daniela Rus, and Song Han. Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2023. 15, 16

[33] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. 15

[34] Emanuele Malagoli and Luca Di Persio. 2d object detection: A survey. *Mathematics*, 13(6), 2025. 15

[35] 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. 23

[36] Edoardo Palladin, Samuel Brucker, Filippo Ghilotti, Praveen Narayanan, Mario Bijelic, and Felix Heide. Self-supervised sparse sensor fusion for long range perception. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025. 23

[37] Yue Pan, Xingguang Zhong, Louis Wiesmann, Thorbjörn Posewsky, Jens Behley, and Cyrill Stachniss. Pin-slam: Lidar slam using a point-based implicit neural representation for achieving global map consistency. *IEEE Transactions on Robotics*, 40:4045–4064, 2024. 4, 9

[38] Luigi Piccinelli, Christos Sakaridis, Yung-Hsu Yang, Mattia Segu, Siyuan Li, Wim Abbeloos, and Luc Van Gool. UniDepthV2: Universal monocular metric depth estimation made simpler, 2025. 19, 22, 24

[39] Matthias Muller René Reiner Birkl, Diana Wofk. Midas v3.1 – a model zoo for robust monocular relative depth estimation. *CoRR*, abs/2307.14460, 2021. 22

[40] Aron Schmied, Tobias Fischer, Martin Danelljan, Marc Pollefeys, and Fisher Yu. R3d3: Dense 3d reconstruction of dynamic scenes from multiple cameras. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3216–3226, 2023. 19, 20

[41] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 24

[42] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 Conference Proceedings*, 2023. 24

[43] Zachary Teed and Jia Deng. Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras. *Advances in neural information processing systems*, 34:16558–16569, 2021. 19

[44] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. Sparsity invariant cnns. In *International Conference on 3D Vision (3DV)*, 2017. 13, 21

[45] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5294–5306, 2025. 20

[46] Qitai Wang, Yuntao Chen, Ziqi Pang, Naiyan Wang, and Zhaoxiang Zhang. Immortal tracker: Tracklet never dies. *arXiv preprint arXiv:2111.13672*, 2021. 18

[47] 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. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS Datasets and Benchmarks 2021)*, 2021. 15

[48] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. In *Neural Information Processing Systems (NeurIPS)*, 2021. 23

[49] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 2018. 25

[50] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. 4

[51] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. *arXiv:2406.09414*, 2024. 12

[52] 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*, 2024. 22, 23

[53] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3d object detection and tracking, 2021. 18

[54] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M. Ni, and Heung-Yeung Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection, 2022. 14, 15, 16

[55] Tianyuan Zhang, Xuanyao Chen, Yue Wang, Yilun Wang, and Hang Zhao. Mutr3d: A multi-camera tracking framework via 3d-to-2d queries. *arXiv preprint arXiv:2205.00613*, 2022. 18

[56] Hongyu Zhou, Jiahao Shao, Lu Xu, Dongfeng Bai, Weichao Qiu, Bingbing Liu, Yue Wang, Andreas Geiger, and Yiyi Liao. Hugs: Holistic urban 3d scene understanding via gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21336–21345, 2024. 24