

1243 Appendix A Extended Related Work

1244 This section expands the brief positioning given in the main text, covering both
1245 LiDAR semantic segmentation architectures and the main paradigms for handling
1246 cross-LiDAR domain shift.

1248 A.1 LiDAR-based Semantic Segmentation

1250 LiDAR semantic segmentation assigns a semantic label to each point in a LiDAR
1251 scan, enabling autonomous vehicles to perceive and understand their surroundings.
1252 Depending on the form of representation, existing methods can be broadly grouped
1253 into *point-based*, *projection-based*, and *voxel-based* approaches. **Point-based** methods
1254 directly process raw point clouds and were among the earliest solutions [16, 33–39];
1255 although intuitive, they often struggle to capture structured local context efficiently,
1256 leading to suboptimal performance. **Projection-based** methods transform point
1257 clouds into 2D representations such as range images or bird’s-eye views, so that mature
1258 2D vision architectures can be applied [18, 40–43], but projection inevitably intro-
1259 duces information loss and geometric distortion. **Voxel-based** methods discretize 3D
1260 space into voxels and explicitly model local geometric relations [17, 44–50]; they gen-
1261 erally achieve stronger performance, but their efficiency and accuracy are inherently
1262 constrained by voxel resolution.

1263 Despite these differences, most of these methods follow the SDT paradigm
1264 and therefore tend to overfit the sampling pattern of a specific LiDAR, degrading
1265 severely when deployed on unseen sensors—which highlights the need for cross-LiDAR
1266 generalization.

1268 A.2 Transfer Learning for LiDAR

1270 Efforts to address LiDAR domain shift generally fall into three lines. **Domain adap-**
1271 **tation** methods [3, 24, 51–54] reduce feature-distribution discrepancies by adapting
1272 a model to a specific target LiDAR. While effective for source-to-target transfer, they
1273 require target-domain access with retraining and often suffer from catastrophic for-
1274 getting [6], which limits their applicability in zero-shot scenarios. **Multi-dataset**
1275 **training** methods [7, 8] attempt to construct a unified model across multiple LiDARs,
1276 but their robustness remains bounded by the predefined training distribution and does
1277 not naturally extend to unseen LiDAR configurations.

1278 Recent efforts on **cross-LiDAR generalization** can be divided into two cat-
1279 egories according to how they address domain shift. The first improves robustness
1280 through **feature regularization under restricted domain variation** [9–12], using
1281 strategies such as beam dropping, density-aware modeling, BEV priors, and cross-
1282 modal supervision. These methods improve feature robustness under specific forms of
1283 variation, but do not explicitly model the LiDAR sampling process itself; as a result,
1284 they mainly address restricted changes such as density or sparsity, rather than how
1285 sensor-specific sampling distorts the observation of the same underlying scene. The sec-
1286 ond improves generalization through **geometry reconstruction** [13, 14, 24], seeking
1287 a more complete geometric representation beyond the raw sparse scan. Among them,
1288

3DLabelProp [13] improves robustness through multi-frame accumulation and geometric label propagation, so its gains mainly rely on sequence-level aggregation rather than explicit modeling of cross-sensor sampling variation. LiDomAug [14] augments training data by rendering observations under different cylindrical LiDAR configurations from a multi-frame accumulated scene, yet its generalization is still driven by augmentation over a set of predefined sensor configurations. More fundamentally, neither category provides an explicit mechanism to separate “SIG” from sensor-induced “SG”, and thus cannot directly steer the model toward sensor-invariant scene structure.

In contrast, LiSeer formulates cross-LiDAR generalization by treating different LiDARs as different sparse samplers of the same underlying 3D world. It learns from randomized and controllable sampling geometries and explicitly decouples “SIG” from “SG”, so that the model is guided to focus on the former rather than overfit the latter.

Appendix B More Analysis for Sampling Module

B.1 Parameter settings

Rather than drawing from the full parameter space throughout training, we adopt a *curriculum* strategy for virtual LiDAR simulation: the sampling ranges are progressively widened across training stages, so that the model is first exposed to simpler, less diverse sensor configurations before being optimized over broader and more challenging settings. This curriculum encourages the learning of more intrinsic, sensor-invariant representations and improves generalization to unseen LiDAR domains.

At each stage, we uniformly sample the five configuration parameters introduced in the main text—the number of vertical beams, the horizontal angular resolution, the lower and upper bounds of the vertical field of view, and the installation height. As summarized in Table B1, these ranges are enlarged stage by stage following the curriculum.

Table B1: Stage-wise curriculum for simulating diverse LiDAR configurations.

Stage (Epochs)	Beams	Height	V-L-Bound	V-H-Bound	H-resolution
1 ([0, 8))	[32, 64]	[1.7, 1.9] m	$[-30, -20]^\circ$	$[0, 10]^\circ$	[900, 2400]
2 ([8, 16))	[24, 72]	[1.5, 2.0] m	$[-35, -15]^\circ$	$[0, 15]^\circ$	[600, 3000]
3 ([16, 24))	[16, 80]	[1.2, 2.0] m	$[-35, -10]^\circ$	$[0, 20]^\circ$	[400, 4000]

Note. V-L-Bound and V-H-Bound denote the lower and upper bounds of the vertical field of view, respectively. H-resolution denotes the horizontal angular resolution. The curriculum progressively enlarges the sampling ranges from Stage 1 to Stage 3.

B.2 Distortion-aware scanline regularization.

Even after visibility filtering, directly retaining the raw coordinates of the selected points can still produce unrealistically thick scanline bands. The cause is that discretized ray assignment introduces depth-quantization noise: once projected back into

1335 3D, the points no longer lie on the ideal conical scan manifold of a mechanical LiDAR.
 1336 This effect grows more pronounced at long range, producing visible banding artifacts.
 1337 To restore physically plausible scanlines, we introduce a distance-adaptive ring
 1338 regularization. For each beam, we estimate the local geometric trend along azimuth
 1339 using an edge-preserving circular median filter,

$$1340 \quad \tilde{r}_{u,v} = \text{Median}(\{r_{u+\delta,v}\}_{\delta=-W_s}^{W_s}),$$

1343 where $r_{u,v}$ denotes the depth at azimuth index u and beam index v . Within geomet-
 1344 rically continuous regions, the observed depth is softly contracted toward this local
 1345 trend:

$$1346 \quad \hat{r}_{u,v} = (1 - \alpha(r_{u,v})) r_{u,v} + \alpha(r_{u,v}) \tilde{r}_{u,v},$$

1347 where the absorption weight $\alpha(r_{u,v}) \in [0, 1]$ increases with distance, reflecting the
 1348 fact that angular-quantization errors become more severe at long range. Finally, the
 1349 sampled point is snapped back onto the ideal simulated ray,

$$1351 \quad \mathbf{p}'_{u,v} = \mathbf{o} + \hat{r}_{u,v} \mathbf{d}_{u,v},$$

1353 where $\mathbf{o}$ is the virtual LiDAR origin and $\mathbf{d}_{u,v}$ is the corresponding unit ray direction.
 1354 This correction suppresses local coordinate scatter and restores the sharp scanline
 1355 structure of real LiDAR measurements.

1356 In summary, this sampling module goes beyond conventional data augmentation:
 1357 by sampling directly in 3D under randomized, controllable sensor parameters, explic-
 1358 itly accounting for occlusion, and correcting scanline distortion, it exposes the model
 1359 to broad yet physically grounded sampling variation—which is essential for learning
 1360 generalizable scene representations rather than overfitting to a few predefined sensor
 1361 layouts.

1363 B.3 Pseudo-code for the efficient controllable sampling 1364 algorithm

1366 The sampler proceeds in three stages: (i) controllable sensor construction from ran-
 1367 domized LiDAR parameters, (ii) efficient point-wise angular assignment followed by
 1368 per-ray nearest-hit reduction, and (iii) local refinement in scan space. Unlike a naïve
 1369 ray-wise search, this implementation avoids repeated global point comparisons and
 1370 performs all subsequent corrections directly in the discrete scan space. The complete
 1371 procedure is summarized in Algorithm 1.

1373 B.4 Efficiency analysis

1374 Let N denote the number of points in the canonical dense point cloud, B the number
 1375 of vertical beams, and H the number of horizontal angular steps. The total number
 1376 of simulated rays (or scan bins) is therefore

$$1378 \quad R = B \cdot H. \quad (\text{B1})$$

Algorithm 1 Efficient Physics-based Controllable Sampling

Require: Dense point cloud $P = \{\mathbf{p}_i\}_{i=1}^N$, labels $Y = \{y_i\}_{i=1}^N$, sensor parameter ranges

Ensure: Sampled sparse point cloud P' , sampled labels Y'

- 1: // **Stage 0:** controllable sensor construction
- 2: Sample LiDAR parameters $(h, B, H, [\theta_{\min}, \theta_{\max}])$ from predefined ranges
- 3: Construct $R = B \times H$ discrete scan rays $\{\mathbf{r}_j\}_{j=1}^R$
- 4: Set virtual LiDAR origin $\mathbf{o} = (0, 0, h)$
- 5: // **Stage 1:** point-wise angular assignment
- 6: Filter points outside the valid range
- 7: **for** each point $\mathbf{p}_i$ **do**
- 8: Compute relative vector $\mathbf{v}_i = \mathbf{p}_i - \mathbf{o}$, range ρ_i , azimuth ϕ_i , and elevation θ_i
- 9: Quantize (ϕ_i, θ_i) to the nearest angular bin $b_i \in \{1, \dots, R\}$
- 10: Evaluate angular consistency between $\mathbf{v}_i$ and ray $\mathbf{r}_{b_i}$
- 11: Mark $\mathbf{p}_i$ as valid only if it lies within the angular footprint of $\mathbf{r}_{b_i}$
- 12: **end for**
- 13: // **Stage 2:** per-ray nearest-hit reduction
- 14: Initialize depth buffer $D[1..R] \leftarrow +\infty$ and index buffer $A[1..R] \leftarrow -1$
- 15: **for** each valid point $\mathbf{p}_i$ **do**
- 16: $b \leftarrow b_i$
- 17: Compute projected depth $d_i = \mathbf{v}_i^\top \mathbf{r}_b$
- 18: **if** $d_i < D[b]$ **then**
- 19: Atomically update $D[b] \leftarrow d_i$, $A[b] \leftarrow i$
- 20: **end if**
- 21: **end for**
- 22: // **Stage 3:** scan-space local refinement
- 23: Reshape D and A into a $B \times H$ scan-space grid
- 24: Apply local scan-space refinement on fixed-size neighborhoods
- 25: Refine the retained per-ray depths while preserving one return per occupied bin
- 26: // **Stage 4:** output assembly
- 27: **for** each valid occupied bin b **do**
- 28: Retrieve winning source point index $i = A[b]$
- 29: Reproject the refined depth $\hat{d}_b$ to 3D via $\hat{\mathbf{p}}_b = \mathbf{o} + \hat{d}_b \mathbf{r}_b$
- 30: Append $\hat{\mathbf{p}}_b$ and label y_i to P', Y'
- 31: **end for**
- 32: **return** P', Y'

All elementary arithmetic, trigonometric, and indexing operations are treated as $O(1)$.

A naïve ray-centric simulation would cast each ray independently and search over all points to find the nearest return. This requires comparing every ray against all points, resulting in

$$T_{\text{naive}}(N, R) = O(NR), \quad (\text{B2})$$

which is impractical for online training when both the dense point cloud and the scan resolution are large.

1427 In contrast, our implementation decomposes the computation into three efficient
 1428 stages.

1429

1430 **(1) *Point-wise geometric preprocessing and angular assignment.***

1431 Each point is processed only once. Specifically, we compute its relative vector to the
 1432 virtual LiDAR origin, its range for distance filtering, and its angular coordinates
 1433 (azimuth and elevation), and then quantize it directly to its nearest angular bin in
 1434 the discrete $B \times H$ scan grid. Since each point incurs a constant amount of work, the
 1435 total cost of this stage is

$$1436 \quad T_{\text{geom}}(N) = O(N). \quad (\text{B3})$$

1437 Compared with a bin-search formulation, this removes the explicit logarithmic search
 1438 over the angular grid by quantizing each point directly to its nearest bin.

1439

1440 **(2) *Per-ray nearest-hit reduction.***

1441 After angular assignment, each valid point contributes to exactly one scan bin and
 1442 competes only with other points assigned to the same ray. We maintain a depth buffer
 1443 and an index buffer of length R , initialized in

$$1445 \quad T_{\text{init}}(R) = O(R). \quad (\text{B4})$$

1447 A segmented minimum reduction is then performed over all valid points, retaining
 1448 only the closest point for each occupied ray. Because each point participates in exactly
 1449 one bin update, this stage is linear in the number of points:

$$1451 \quad T_{\text{reduce}}(N) = O(N). \quad (\text{B5})$$

1453 Thus, the total cost of the nearest-hit construction stage is

$$1455 \quad T_{\text{zbuf}}(N, R) = O(N + R). \quad (\text{B6})$$

1457

1458 **(3) *Scan-space local refinement.***

1459 The subsequent refinement is no longer performed in the original point space, but
 1460 entirely in the discrete scan space of size $B \times H$. This includes local neighbor-
 1461 hood operations over fixed-size windows, such as scan-space consistency filtering and
 1462 scanline-level regularization. Let w_o and h_o denote the azimuth and elevation extents
 1463 of the local scan-space window, w_r the ring-protection neighborhood width, and w_s
 1464 the ring-regularization window size. The corresponding cost is

$$1465 \quad T_{\text{scan}}(R) = O(R(w_o h_o + w_r + w_s \log w_s)). \quad (\text{B7})$$

1467

1468 In practice, all these window sizes are fixed small constants, so this stage scales linearly
 1469 with the number of scan bins:

$$1470 \quad T_{\text{scan}}(R) = O(R). \quad (\text{B8})$$

1472

Overall complexity.

Combining the above stages, the total complexity of our sampler is

$$\begin{aligned} T_{\text{ours}}(N, R) &= T_{\text{geom}}(N) + T_{\text{zbuf}}(N, R) + T_{\text{scan}}(R) \\ &= O(N) + O(N + R) + O(R) \\ &= O(N + R). \end{aligned} \tag{B9}$$

Retaining the scan-window constants yields the more explicit form

$$T_{\text{ours}}(N, R) = O(N + R(w_o h_o + w_r + w_s \log w_s)). \tag{B10}$$

Since the neighborhood window sizes are fixed and small in all experiments, the practical behavior is near-linear. Under typical LiDAR settings, where $N \gg R$, the runtime is effectively dominated by the point-wise preprocessing stage and behaves approximately as

$$T_{\text{ours}}(N, R) \approx O(N). \tag{B11}$$

Memory complexity.

The dominant memory consumption comes from the point-level tensors of size $O(N)$ and the scan-space buffers of size $O(R)$. The temporary neighborhood tensors introduced by local scan-space unfolding are of size $O(R \cdot w)$, where w is a fixed small window size. Therefore, the overall memory complexity remains

$$M_{\text{ours}}(N, R) = O(N + R). \tag{B12}$$

Practical efficiency.

Compared with the naïve $O(NR)$ ray-centric search, our formulation removes the multiplicative dependence on the number of rays, replacing it with a point-wise angular assignment followed by linear scan-space processing. In practice, we observe no noticeable increase in per-epoch training time after introducing the controllable sampling module, indicating that the simulator is light enough for online use. Any slowdown arises only when extremely large densified point clouds are read from slow mechanical storage, in which case the bottleneck is CPU–GPU I/O rather than the sampling algorithm itself; on solid-state storage this bottleneck disappears.

Appendix C Visualization of Label Propagation

The label-propagation procedure is simple, deterministic, and computationally efficient, requiring only $O(N \log N)$ time thanks to a nearest-voxel search implemented with a k -d tree, as shown in Fig. C1. It therefore scales reliably to large datasets such as SemanticKITTI, yielding dense reconstructions that remain semantically consistent with the original sparse annotations.

Fig. C2 presents qualitative comparisons between the original sparse labels and the propagated dense annotations across nuScenes, SemanticKITTI, and Waymo. Urban structures—including roads, sidewalks, and building facades—retain sharp and

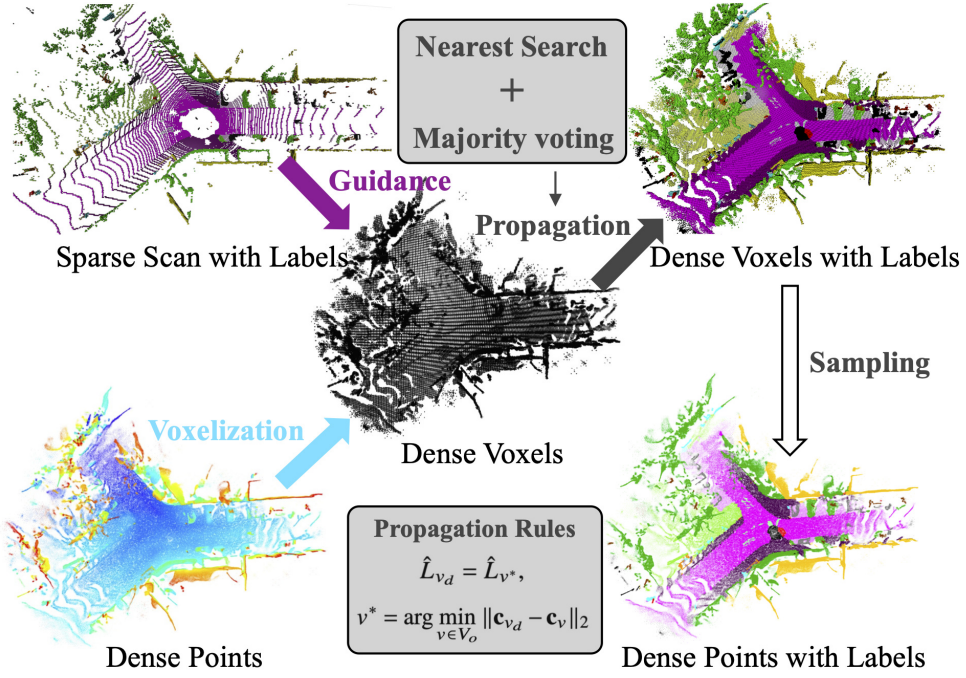


Fig. C1: Illustration for the label propagation module.

coherent boundaries, showing that majority voting within the original voxels provides robust aggregation even in geometrically complex scenes. Dynamic objects such as vehicles and pedestrians also keep well-defined shapes after propagation, indicating that nearest-voxel interpolation introduces no noticeable semantic drift. The effect is particularly clear in scenes with curved roads or occlusions, where newly introduced points inherit labels that align smoothly with the sparse ground truth, without visible artifacts or discontinuities.

The dense reconstructions also improve structural completeness without compromising semantic correctness. Regions only sparsely captured in the original scans—such as upper vegetation layers or fine-grained roadside geometry—become densely populated while retaining labels consistent with their surrounding context. This indicates that the propagation mechanism transfers semantic information from spatially adjacent voxels without over-smoothing or collapsing category boundaries.

A key observation is that the quality of the propagated labels is strongly correlated with the quality and consistency of the underlying sparse annotations. SemanticKITTI, which provides the most precise and spatially coherent labels among the evaluated datasets, yields the cleanest and most stable dense annotations; nuScenes gives moderately consistent results, reflecting its intermediate annotation fidelity; and the propagated labels on Waymo appear noisier and less continuous, chiefly because of the noise already present in its sparse annotations. These differences indicate that the propagation mechanism faithfully inherits the characteristics of the source dataset.

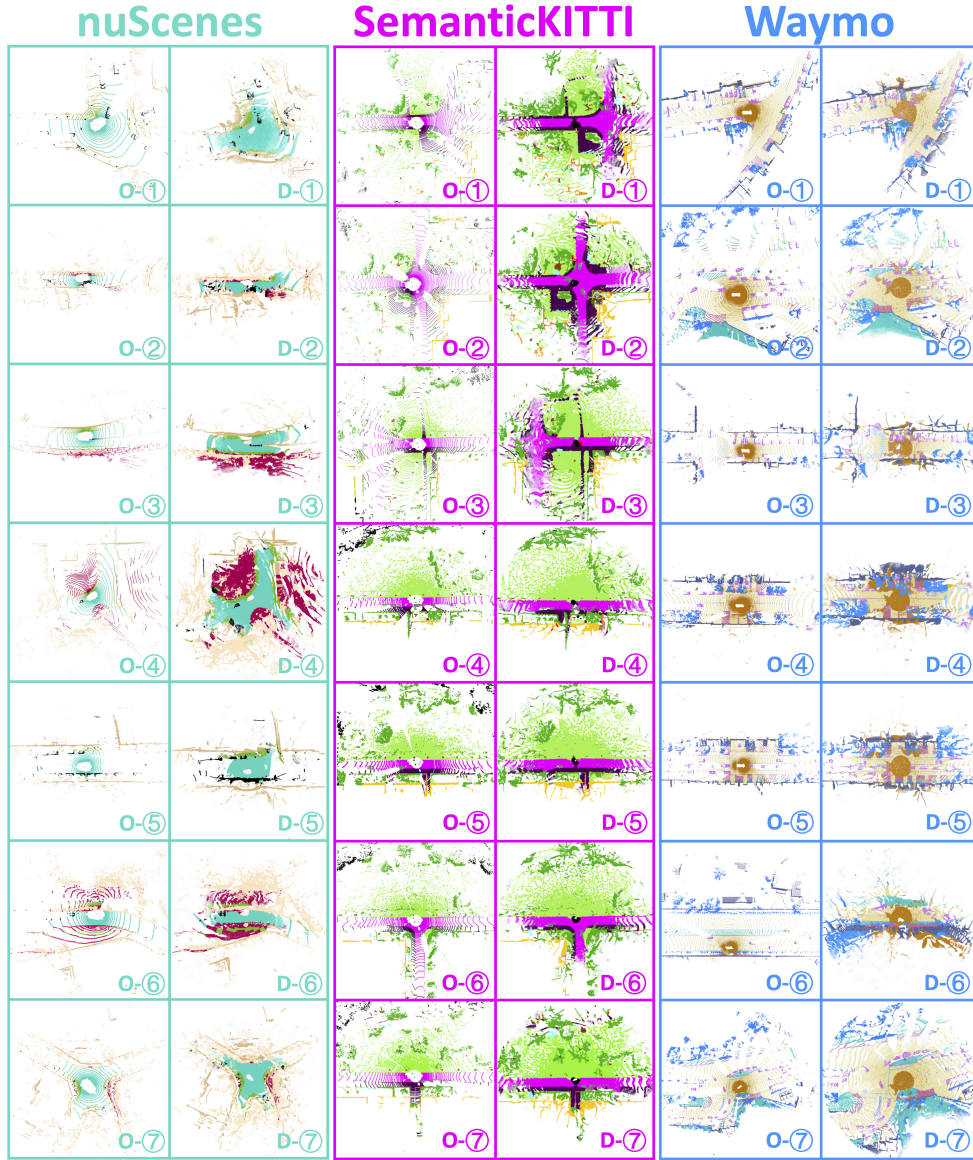


Fig. C2: Label propagation visualization.

Note. Qualitative comparison between sparse input labels and propagated dense labels across nuScenes, SemanticKITTI, and Waymo.

It is also worth noting that the module makes the most semantically plausible inference in regions lacking direct measurements. Although labels assigned to entirely unobserved areas may occasionally deviate from the true semantics, such discrepancies are largely absorbed by the randomness and diversity of the subsequent sampling

process. In practice, as long as labels in unobserved regions preserve local semantic continuity and reasonable fidelity, they are fully adequate for supervising downstream learning on the densified point clouds.

Overall, these results demonstrate that the propagation module produces dense annotations that faithfully reflect the structure and semantics of the original datasets, preserve high-quality category boundaries, and provide reliable supervision for the learning framework.

Appendix D More Details on Coarse Label

We use the coarse labels for all cross-domain (domain-generalization) evaluations; the indices range from 0 to 7. The detailed mapping rules for the three datasets are given in Listings D.1, D.2, and D.3.

Code D.1: Label mapping rules from nuScenes to coarse label.

```

NUSCENES_TO_CL = {
0: 0,    \# void / ignore -> Other
1: 6,    \# barrier -> Objects
2: 6,    \# bicycle -> Objects
3: 3,    \# bus -> Vehicle
4: 3,    \# car -> Vehicle
5: 3,    \# construction\_vehicle -> Vehicle
6: 3,    \# motorcycle -> Vehicle
7: 5,    \# pedestrian -> Living Being
8: 6,    \# traffic\_cone -> Objects
9: 3,    \# trailer -> Vehicle
10: 3,   \# truck -> Vehicle
11: 1,   \# driveable\_surface -> Driveable Ground
12: 7,   \# other\_flat -> Other Ground
13: 7,   \# sidewalk -> Other Ground
14: 4,   \# terrain -> Nature
15: 2,   \# manmade -> Structure
16: 4,   \# vegetation -> Nature
}
```

Code D.2: Label mapping rules from SemanticKITTI to coarse label.

```

SEMANTICKITTI_TO_CL = {
0: 0,    \# unlabeled -> Other
1: 0,    \# outlier -> Other
10: 3,   \# car -> Vehicle
11: 3,   \# bicycle -> Vehicle
13: 3,   \# bus -> Vehicle
15: 3,   \# motorcycle -> Vehicle
16: 3,   \# on-rails -> Vehicle
18: 3,   \# truck -> Vehicle
}
```

```

10      20: 3,    \# other-vehicle -> Vehicle
11      30: 5,    \# person -> Living Being
12      31: 5,    \# bicyclist -> Living Being
13      32: 5,    \# motorcyclist -> Living Being
14      40: 1,    \# road -> Driveable Ground
15      44: 1,    \# parking -> Driveable Ground
16      48: 7,    \# sidewalk -> Other Ground
17      49: 7,    \# other-ground -> Other Ground
18      50: 2,    \# building -> Structure
19      51: 2,    \# fence -> Structure
20      52: 2,    \# other-structure -> Structure
21      60: 1,    \# lane-marking -> Driveable Ground
22      70: 4,    \# vegetation -> Nature
23      71: 4,    \# trunk -> Nature
24      72: 4,    \# terrain -> Nature
25      80: 6,    \# pole -> Objects
26      81: 6,    \# traffic-sign -> Objects
27      99: 0,    \# other-object -> Other
28      252: 6,   \# moving-car -> Objects
29      253: 3,   \# moving-bicyclist -> Vehicle (treated as vehicle
                for DG)
30      254: 5,   \# moving-person -> Living Being
31      255: 5,   \# moving-motorcyclist -> Living Being
32      256: 6,   \# moving-on-rails -> Objects
33      257: 6,   \# moving-bus -> Objects
34      258: 6,   \# moving-truck -> Objects
35      259: 6,   \# moving-other-vehicle -> Objects
36  }

```

Code D.3: Label mapping rules from Waymo to coarse label.

```

1  WAYMO_TO_CL = {
2      0: 0,    \# TYPE\_UNDEFINED -> Other
3      1: 3,    \# TYPE\_CAR -> Vehicle
4      2: 3,    \# TYPE\_TRUCK -> Vehicle
5      3: 3,    \# TYPE\_BUS -> Vehicle
6      4: 3,    \# TYPE\_OTHER\_VEHICLE -> Vehicle
7      5: 5,    \# TYPE\_MOTORCYCLIST -> Living Being
8      6: 5,    \# TYPE\_BICYCLIST -> Living Being
9      7: 5,    \# TYPE\_PEDESTRIAN -> Living Being
10     8: 6,    \# TYPE\_SIGN -> Objects
11     9: 6,    \# TYPE\_TRAFFIC\_LIGHT -> Objects
12    10: 6,    \# TYPE\_POLE -> Objects
13    11: 6,    \# TYPE\_CONSTRUCTION\_CONE -> Objects
14    12: 6,    \# TYPE\_BICYCLE -> Objects
15    13: 6,    \# TYPE\_MOTORCYCLE -> Objects
16    14: 2,    \# TYPE\_BUILDING -> Structure

```



```

1703 15: 4, \# TYPE\_VEGETATION -> Nature
1704 16: 4, \# TYPE\_TREE\_TRUNK -> Nature
1705 17: 1, \# TYPE\_CURB -> Driveable Ground
1706 18: 1, \# TYPE\_ROAD -> Driveable Ground
1707 19: 1, \# TYPE\_LANE\_MARKER -> Driveable Ground
1708 20: 7, \# TYPE\_OTHER\_GROUND -> Other Ground
1709 21: 7, \# TYPE\_WALKABLE -> Other Ground
1710 22: 7, \# TYPE\_SIDEWALK -> Other Ground
1711 }

```

Appendix E More Visualization of Densified Point Cloud

Dense point clouds are also visualized in Fig. C2 for nuScenes, SemanticKITTI, and Waymo. Each column compares the original sparse annotations (“O-#”) with the propagated dense annotations (“D-#”) for the same scene.

For nuScenes, the annotation files do not follow a consistent naming correspondence with the point-cloud files, so we randomly sampled several files from the full dataset for visualization. For SemanticKITTI, we randomly selected eight frames from sequence 08 for display: 000000.bin, 000050.bin, 000100.bin, 000150.bin, 000200.bin, 000250.bin, and 000300.bin. For Waymo, because only a subset of LiDAR scans carries point-wise semantic labels, we present the first annotated frames available in the dataset. Together, these examples further illustrate the visual consistency, semantic continuity, and structural completeness of the densified representations across datasets.

Appendix F Qualitative Results and Analysis of On-Vehicle Testing on Unseen LiDAR Platforms

To further assess the practical value of LiSeer beyond standard benchmarks, we conduct real-vehicle validation on three in-house platforms equipped with completely unseen LiDARs. The goal of this experiment is to test whether the proposed zero-shot cross-LiDAR framework can transfer to real deployment platforms without any target-domain adaptation or fine-tuning. Since the collected real-world data do not provide dense point-wise ground truth, we evaluate qualitatively by comparing the projected LiDAR segmentation against synchronized camera images; to this end, each platform carries a forward-facing camera in addition to its LiDAR.

The real-world data are collected in three representative environments in Beijing: (1) the Tsinghua University campus in Haidian District, (2) the Beijing High-Level Autonomous Driving Demonstration Zone in Yizhuang, and (3) a rural road in Shunyi District. These scenes cover campus roads, urban roads, intersections, vegetation, parked vehicles, roadside structures, and open rural areas, providing diverse conditions for evaluating real-world cross-LiDAR transfer.

For each platform, we visualize the camera image alongside the predictions of the baseline PointSeg model and LiSeer-P. In every qualitative figure, the leftmost column shows the camera image, and the remaining six columns show Baseline-PointSeg and LiSeer-P trained on nuScenes, SemanticKITTI, and Waymo, respectively.

A consistent pattern emerges across all three real platforms. Models trained under the conventional SDT paradigm do not generalize reliably to unseen LiDARs, and their failure modes differ markedly with the source dataset. The nuScenes-trained baseline often preserves coarse road structure, yet its predictions are unstable on vehicles and roadside objects, especially under denser unseen scans. The SemanticKITTI-trained baseline degrades the most severely, frequently collapsing large portions of the scene into the *Other* category—a clear sign of strong sensitivity to sensor-dependent sampling. The Waymo-trained baseline is generally more stable than the SemanticKITTI-trained one, but still shows dataset-specific bias, for example over-expanding certain structural or vegetation regions and producing inconsistent object boundaries across platforms. Under SDT training, then, the model does not fail uniformly; it inherits different dataset-specific biases and expresses them differently when facing unseen LiDAR configurations.

In contrast, LiSeer-P delivers markedly more consistent predictions across all three unseen sensors, regardless of whether it is trained on nuScenes, SemanticKITTI, or Waymo. Despite the substantial gap between the benchmark datasets and the real-world scenes collected in China, LiSeer-P still recovers coherent scene semantics on every platform. In particular, it identifies drivable areas, vehicles, vegetation/nature regions, and roadside objects more reliably than the corresponding baselines, while preserving a clearer large-scale scene layout. This cross-dataset consistency is especially telling: it indicates that the framework is not merely overfitting to a particular source dataset, but learning representations that remain stable across both unseen sensors and unseen geographic environments.

Some failure cases nonetheless remain. In extremely sparse or distant regions, predictions can become fragmented for lack of geometric support; fine-grained object boundaries may be imperfect when the camera-visible structure is only weakly sampled by the LiDAR; and in highly cluttered scenes, small movable objects are occasionally confused with surrounding structure or vegetation when their point support is limited. Even so, compared with the baselines, LiSeer-P exhibits far fewer catastrophic failures and maintains substantially stronger semantic consistency across platforms.

F.1 TH-CS55 platform

TH-CS55 is an in-house passenger-vehicle platform equipped with an unseen 128-beam LiDAR and a forward-facing camera. It represents a substantially denser LiDAR configuration than the benchmark sensors and is therefore particularly useful for testing whether the learned representations remain stable under previously unseen high-density sampling.

On TH-CS55, the gap between the baselines and LiSeer-P is particularly clear. The SemanticKITTI-trained baseline shows the strongest collapse, with many points incorrectly mapped to *Other*, especially in rows 1, 3, 5, and 8 of Fig. F4. The nuScenes-trained baseline is more stable on drivable areas but often under-segments vehicles and

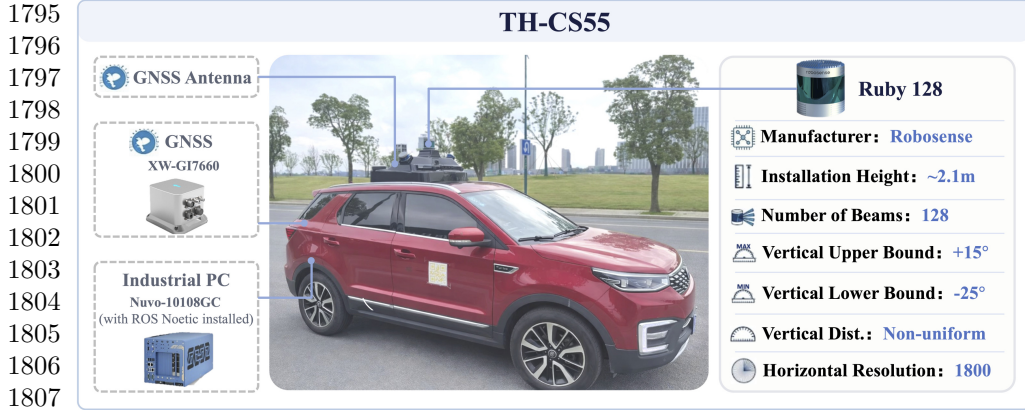


Fig. F3: Overview of the *TH-CS55* platform used for real-vehicle validation.

Note. TH-CS55 is equipped with a front-view camera, a GNSS system, an industrial PC, and an unseen 128-beam LiDAR mounted on the roof. This platform allows us to evaluate cross-LiDAR transfer under a substantially denser sampling configuration than those used in the benchmark datasets.

yields less coherent object regions. The Waymo-trained baseline captures large-scale structure more reliably, yet still shows inconsistent object extents and category transitions under the dense 128-beam observations. By contrast, LiSeer-P remains far more stable across all three source datasets, consistently recovering the drivable surface, nearby vehicles, roadside vegetation, and large roadside objects with better semantic coherence. Even in challenging scenes such as intersections and roadside parking areas, its predictions remain visually plausible and substantially more consistent than those of the corresponding baselines.

F.2 TH-AVT platform

TH-AVT is an in-house autonomous-vehicle platform equipped with an unseen 80-beam LiDAR and a forward-facing camera. Relative to the benchmark sensors used in training, it introduces a dense and previously unseen sampling pattern, making it a challenging test case for zero-shot cross-LiDAR generalization.

The TH-AVT results further highlight the dataset-specific bias of SDT-trained models. The nuScenes-trained baseline tends to preserve the drivable region and some roadside structure, but its vehicle predictions are often incomplete or unstable. The SemanticKITTI-trained baseline again over-predicts *Other*, especially in rows 1, 2, 4, 6, and 7 of Fig. F6, which substantially degrades semantic usability. The Waymo-trained baseline is visually cleaner, but still shows noticeable inconsistency in object boundaries and local semantic transitions. In comparison, LiSeer-P produces much more stable predictions across all three training sources, capturing road layout, building and wall structures, parked vehicles, and roadside vegetation with higher consistency—even though the scenes are collected in Beijing and differ significantly from the original benchmark domains. These results suggest that LiSeer-P is less sensitive to the specific source dataset and more robust to unseen platform geometry and installation conditions.

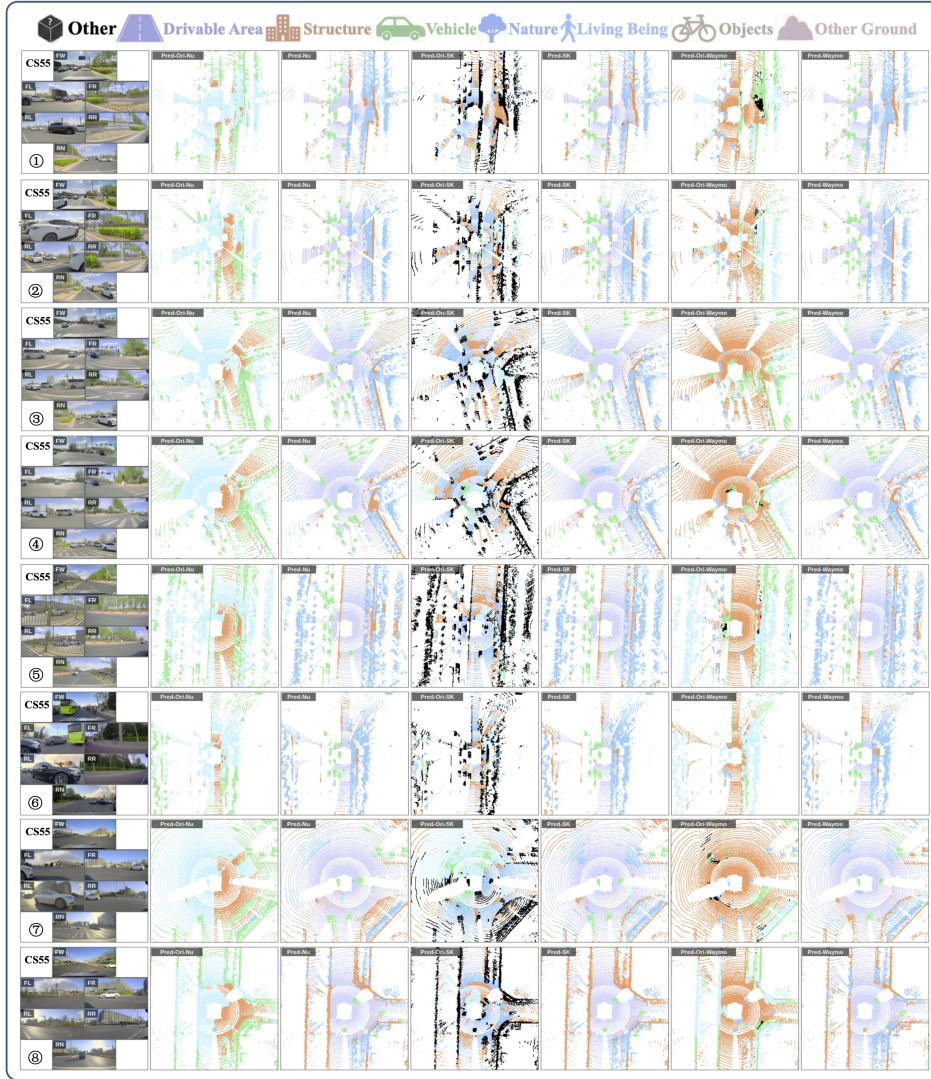


Fig. F4: Qualitative segmentation results on the *TH-CS55* platform.

Note. From left to right: camera image, Baseline-PointSeg and LiSeer-P trained on nuScenes, Baseline-PointSeg and LiSeer-P trained on SemanticKITTI, and Baseline-PointSeg and LiSeer-P trained on Waymo.

F.3 TH-Courier platform

TH-Courier is an in-house compact autonomous courier platform equipped with an unseen 32-beam LiDAR and a forward-facing camera. Unlike TH-AVT and TH-CS55,



Fig. F5: Overview of the *TH-AVT* platform used for real-vehicle validation.

Note. TH-AVT is equipped with a front-view camera, a GNSS system, an industrial PC, and an unseen 80-beam LiDAR mounted at a relatively high position on the vehicle roof. This platform is used to evaluate transferability under a dense and previously unseen LiDAR configuration in real driving scenarios.

it uses a sparser LiDAR mounted at a lower height, providing a complementary deployment scenario for assessing the robustness of LiSeer across substantially different platform geometries.

The TH-Courier platform offers a complementary sparse-LiDAR setting. Here the nuScenes-trained baseline often retains a reasonable estimate of the drivable region, but its predictions on vehicles and roadside objects remain weak and fragmented. The SemanticKITTI-trained baseline again shows the most obvious failure mode, frequently assigning large portions of the scene to *Other*, particularly in rows 1, 3, 6, and 8 of Fig. F8. The Waymo-trained baseline is comparatively more structured, but still exhibits unstable object extents and category boundaries. In contrast, LiSeer-P remains consistently effective across all three source datasets despite the lower mounting height and sparser 32-beam observations, recovering the road surface, vehicles, vegetation, and roadside objects more reliably and preserving a clearer large-scale scene organization in both urban-road and rural-road scenes.

Overall, the results on TH-CS55, TH-AVT, and TH-Courier support two observations. First, conventional SDT-trained models do not generalize reliably to unseen LiDARs, and their failure patterns are strongly source-dataset dependent. Second, LiSeer-P behaves far more consistently across all three unseen sensors and all three source datasets: despite being trained on benchmark data collected in other countries and under different sensor configurations, it transfers effectively to real-world scenes in China and continues to identify drivable surfaces, vehicles, vegetation, and roadside objects with substantially improved consistency. These findings provide direct qualitative evidence for the real-world value of LiSeer in zero-shot cross-LiDAR deployment.

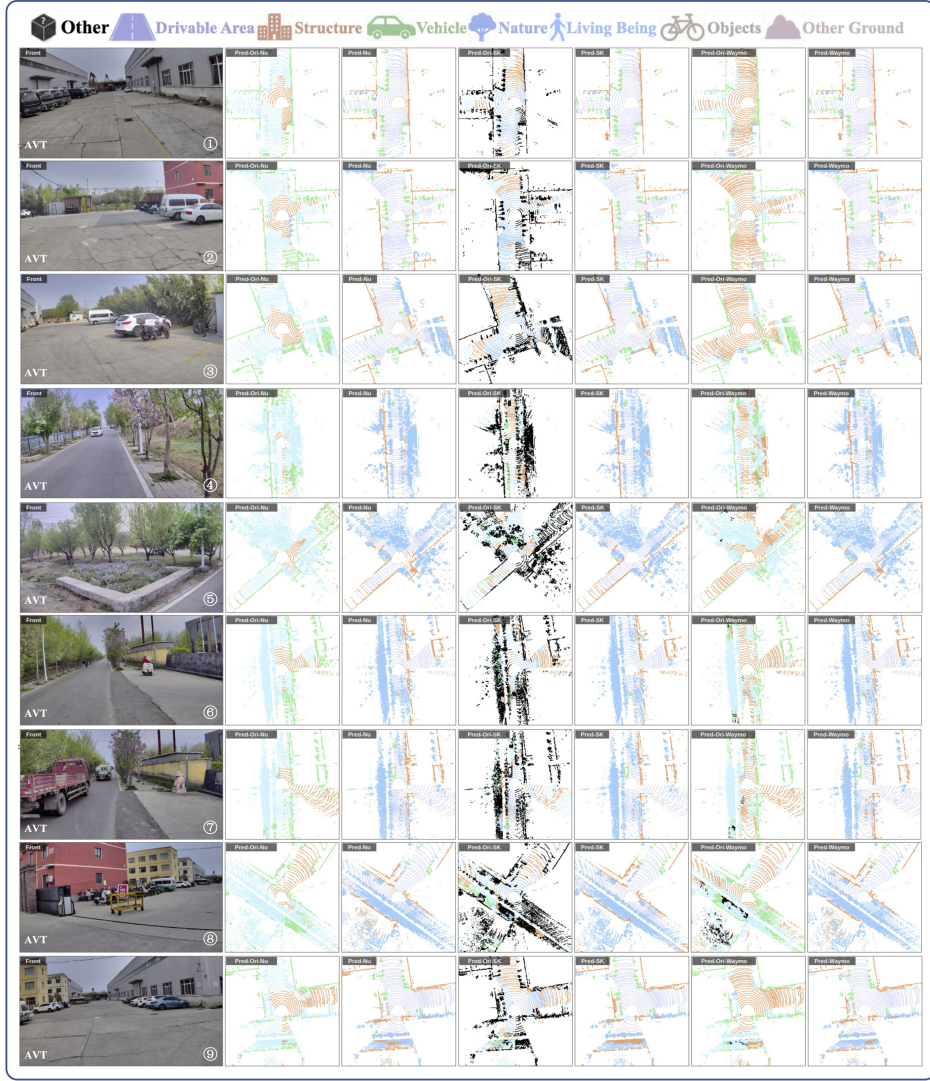


Fig. F6: Qualitative segmentation results on the *TH-AVT* platform.

Note. From left to right: camera image, Baseline-PointSeg and LiSeer-P trained on nuScenes, Baseline-PointSeg and LiSeer-P trained on SemanticKITTI, and Baseline-PointSeg and LiSeer-P trained on Waymo.



Fig. F7: Overview of the *TH-Courier* platform used for real-vehicle validation.

Note. TH-Courier is equipped with a front-view camera, a GNSS system, an industrial PC, and an unseen 32-beam LiDAR mounted at a relatively low position. Owing to its compact platform geometry and sparse LiDAR configuration, it provides a challenging real-world test case that differs substantially from the benchmark training setups.

Appendix G More Settings and Results for Experiments

G.1 Experimental Settings

G.1.1 Model Architecture

As described in Section 2, we adopt two representative baselines: the 3D voxel-based method Cylinder3D, whose backbone is primarily composed of sparse convolution layers, and the 2D projection-based method PointSeg, which employs a Swin Transformer backbone. The input point clouds are partitioned cylindrically into $(\mathcal{H}_{\text{in}}, \mathcal{W}_{\text{in}}, \mathcal{D}_{\text{in}}) = (480, 360, 32)$ voxels, corresponding to radius, angle, and height, respectively.

To ensure a fair comparison, we do not modify any model parameters: all architectural designs and hyper-parameter settings are kept exactly as in the official baselines. Moreover, since pretrained checkpoints are unavailable, all reported results are obtained from our own re-implementations.

G.1.2 Metric for Semantic Segmentation

For 3D semantic segmentation, we evaluate performance using the mean Intersection-over-Union (**mIoU**) across all semantic classes:

$$\text{IoU} = \frac{TP}{TP + FP + FN}$$

$$\text{mIoU} = \frac{1}{C} \sum_{i=1}^C \frac{TP_i}{TP_i + FP_i + FN_i} \quad (\text{G13})$$

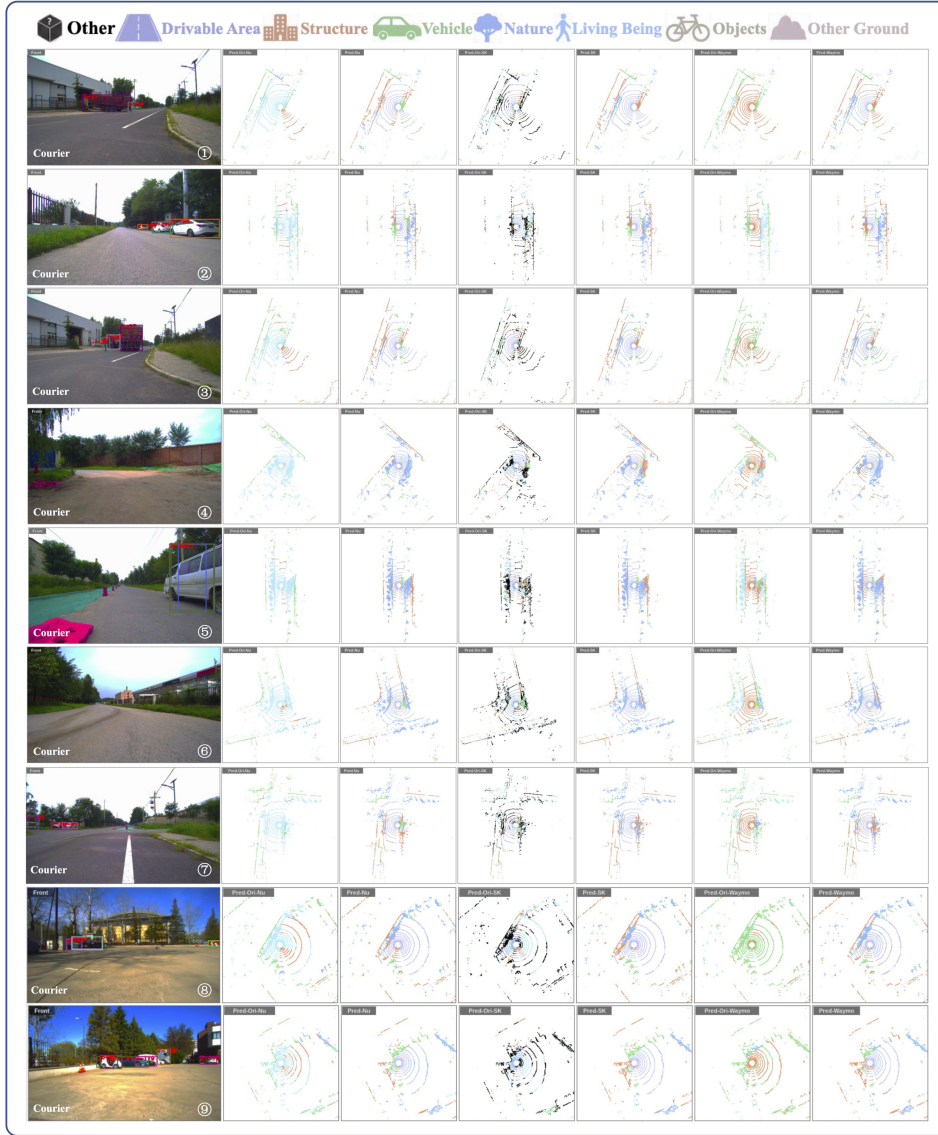


Fig. F8: Qualitative segmentation results on the *TH-Courier* platform.

Note. From left to right: camera image, Baseline-PointSeg and LiSeer-P trained on nuScenes, Baseline-PointSeg and LiSeer-P trained on SemanticKITTI, and Baseline-PointSeg and LiSeer-P trained on Waymo.

where TP , FP , and FN denote the numbers of true positives, false positives, and false negatives, and C denotes the total number of classes.

2071 **G.1.3 Optimization, Training, and Testing**

2072 Across all experiments, we retain the original hyper-parameter configurations of
2073 both baselines without any modification, which further highlights the robustness and
2074 general applicability of our method.

2075 Training uses the Adam optimizer with a weight decay of 0.01. A cosine learning-
2076 rate schedule is adopted with a maximum learning rate of 3×10^{-4} , preceded by
2077 a linear warm-up over the first 500 iterations. The loss function combines standard
2078 cross-entropy with the Lovász-Softmax loss [55], applied jointly at the voxel and point
2079 levels.

2080 All experiments are run within the mmdetection3d framework. Models are trained
2081 for 24 epochs, with a batch size of 4 for Cylinder3D and 1 for PointSeg, distributed
2082 across 4 NVIDIA A800 GPUs.

2084 **G.2 Class-wise Semantic Segmentation Results**

2085
2086 For a more comprehensive view, the class-wise breakdown of the mIoU results reported
2087 in Table 1 is presented in Table G2. As noted in Section 2, the coarse label set reduces
2088 category granularity, and both the $S \rightarrow S$ and $S \rightarrow U$ evaluations exhibit pronounced
2089 long-tail effects that lead to imbalanced per-class performance. For more intuitive
2090 comparison, we additionally plot line and bar charts for all twelve methods on each
2091 dataset, as shown in Fig. G9, Fig. G10, and Fig. G11.

Table G2: Class-wise semantic segmentation results (mIoU, %).

Method	mIoU	Driveable Ground	Structure	Vehicle	Nature	Living Being	Objects	Other Ground
V – baseline – C [†] (SK → SK)	61.7	79.5	83.5	76.4	88.0	14.0	24.5	65.7
P – baseline – C [†] (SK → SK)	71.6	90.7	84.9	92.7	89.9	25.4	43.4	74.4
LiSeer-V-RSCL (SK → SK)	64.3	81.8	77.6	75.6	81.5	37.1	32.5	64.1
LiSeer-P-RSCL (SK → SK)	68.1	86.0	74.9	84.0	81.9	41.8	50.8	57.3
V – baseline – C [†] (SK → NU)	18.1	60.2	24.2	10.3	22.8	0.4	0.0	8.6
P – baseline – C [†] (SK → NU)	19.4	43.3	40.3	2.0	34.7	0.6	0.1	15.0
LiSeer-V-RSCL (SK → NU)	43.2	69.3	60.2	43.1	58.3	22.6	20.3	28.7
LiSeer-P-RSCL (SK → NU)	56.8	70.8	67.7	70.7	63.2	32.8	40.4	51.7
V – baseline – C [†] (SK → Waymo)	10.3	13.5	7.5	11.1	9.0	12.1	6.9	12.2
P – baseline – C [†] (SK → Waymo)	17.3	42.9	29.3	0.8	28.6	0.4	1.6	17.2
LiSeer-V-RSCL (SK → Waymo)	41.3	59.1	59.9	40.4	48.2	20.4	29.3	31.9
LiSeer-P-RSCL (SK → Waymo)	54.6	66.4	77.3	50.8	72.7	22.1	32.9	55.1
V – baseline – C [†] (NU → SK)	8.5	25.9	17.8	7.6	4.4	0.2	0.3	3.5
P – baseline – C [†] (NU → SK)	20.5	29.1	46.0	5.5	50.2	7.6	0.1	4.7
LiSeer-V-RSCL (NU → SK)	36.3	45.3	42.5	45.6	43.3	20.8	23.3	33.5
LiSeer-P-RSCL (NU → SK)	54.4	66.3	62.6	66.7	63.6	33.7	37.0	50.7
V – baseline – C [†] (NU → NU)	56.2	85.9	70.8	69.8	68.1	42.5	1.4	54.6
P – baseline – C [†] (NU → NU)	70.4	90.0	84.6	89.1	82.8	57.5	28.6	60.1
LiSeer-V-RSCL (NU → NU)	57.7	73.0	64.7	63.3	60.7	44.0	47.7	50.5
LiSeer-P-RSCL (NU → NU)	66.5	91.7	84.5	77.9	73.8	43.1	44.1	51.3
V – baseline – C [†] (NU → Waymo)	7.2	10.8	30.7	3.8	3.0	0.2	0.6	1.5
P – baseline – C [†] (NU → Waymo)	18.1	39.3	50.8	4.5	24.0	0.1	0.4	7.8
LiSeer-V-RSCL (NU → Waymo)	33.6	41.9	39.3	42.1	40.0	19.2	21.5	31.3
LiSeer-P-RSCL (NU → Waymo)	53.8	65.3	61.7	65.7	62.7	33.8	37.0	50.5
V – baseline – C [†] (Waymo → SK)	4.0	0.0	1.4	0.9	21.1	0.2	0.3	3.9
P – baseline – C [†] (Waymo → SK)	24.7	42.6	50.0	15.1	41.8	1.0	6.6	15.7
LiSeer-V-RSCL (Waymo → SK)	36.3	45.3	42.5	45.6	43.3	20.8	23.3	33.5
LiSeer-P-RSCL (Waymo → SK)	52.3	66.2	61.9	66.6	63.1	28.2	32.1	48.3
V – baseline – C [†] (Waymo → NU)	3.7	0.0	7.6	0.9	13.0	0.2	0.4	4.0
P – baseline – C [†] (Waymo → NU)	5.4	9.3	19.3	0.7	6.1	0.0	0.0	2.3
LiSeer-V-RSCL (Waymo → NU)	35.0	43.6	40.9	43.8	41.7	20.0	22.4	32.4
LiSeer-P-RSCL (Waymo → NU)	49.0	61.4	57.5	61.7	58.6	26.4	30.9	65.3
V – baseline – C [†] (Waymo → Waymo)	44.0	66.2	67.6	52.7	59.7	5.0	14.3	42.6
P – baseline – C [†] (Waymo → Waymo)	60.9	77.7	77.7	61.7	80.9	33.0	28.0	67.4
LiSeer-V-RSCL (Waymo → Waymo)	44.6	69.2	68.9	49.3	59.7	4.0	13.8	47.0
LiSeer-P-RSCL (Waymo → Waymo)	62.2	72.3	74.6	63.2	72.1	44.0	50.0	58.0

Note. Results correspond to the Source-to-Target setting in Table 1. †: reproduced by the authors. -V / -P: voxel-based / projection-based backbone. -R / -C: physics-based random sampling and coarse label mapping. →: generalization from source to target domain. Highlighted cells follow the same convention as Table 1.

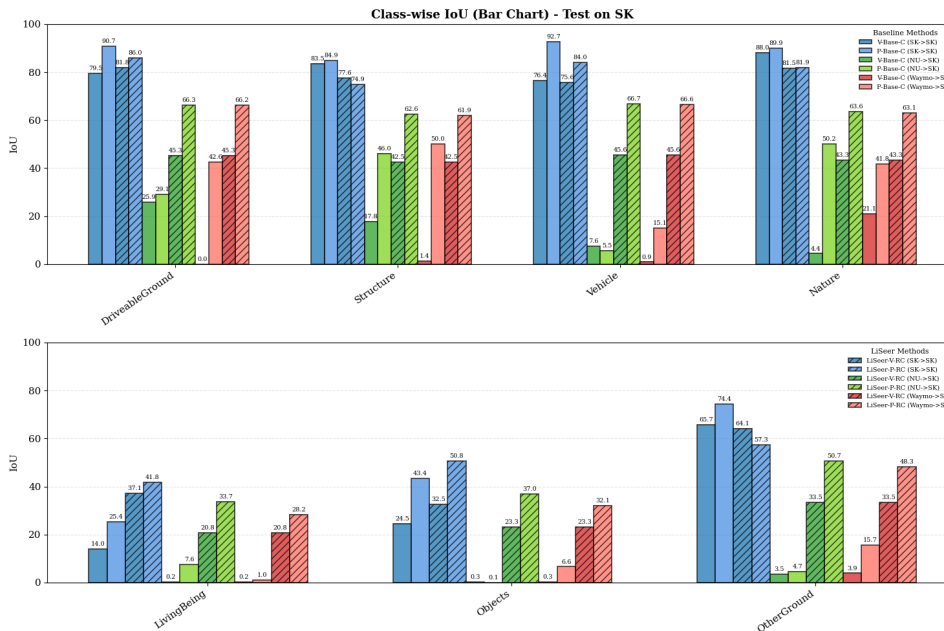
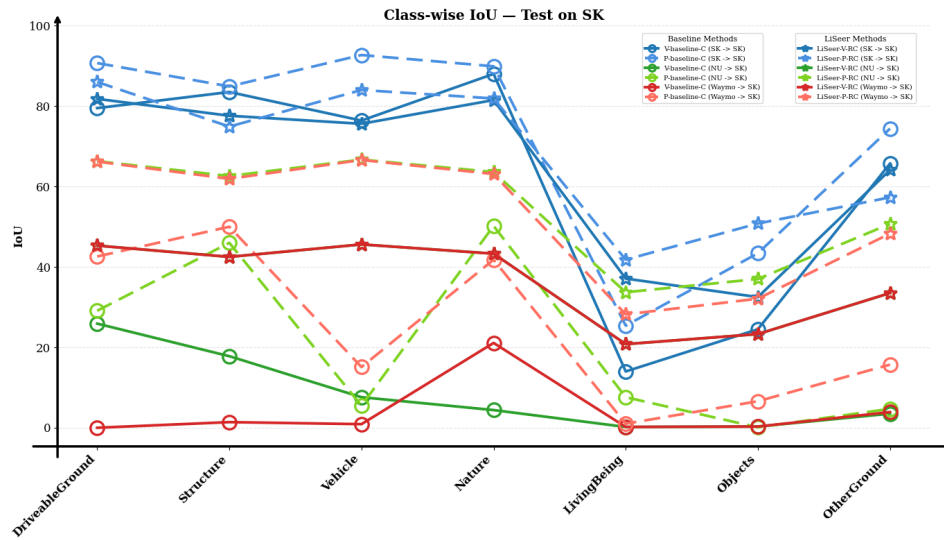
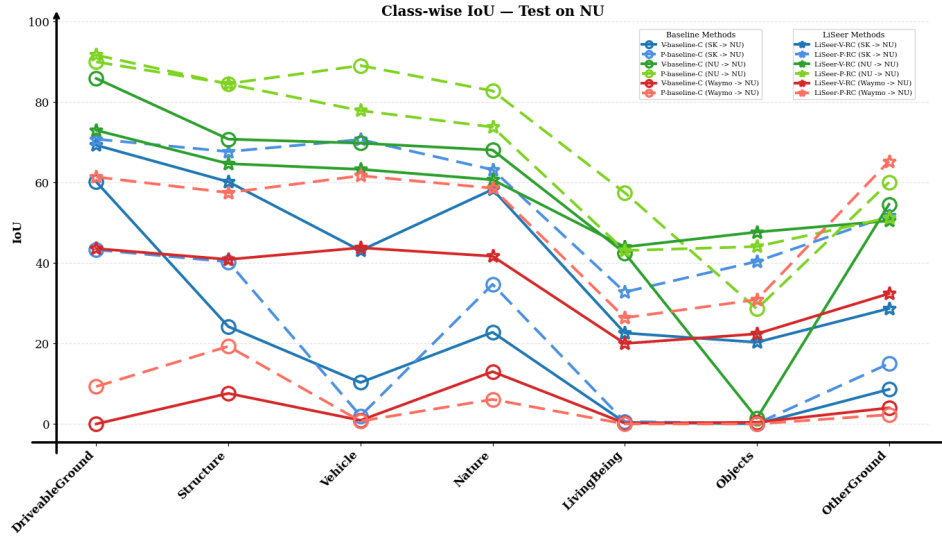
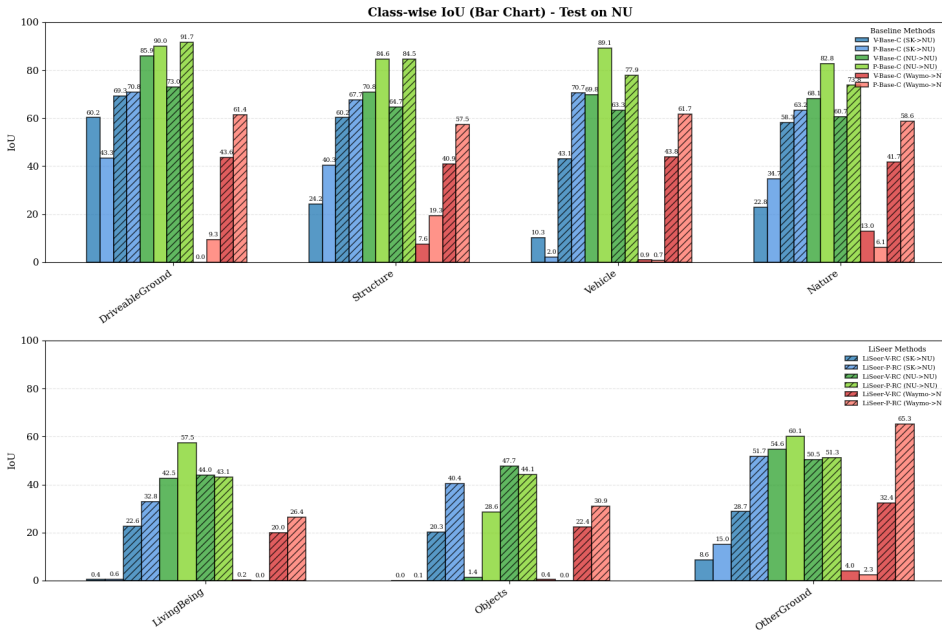


Fig. G9: Class-wise results on SemanticKITTI.

Note. Line and bar charts visualize the same class-wise IoU metrics in different styles.



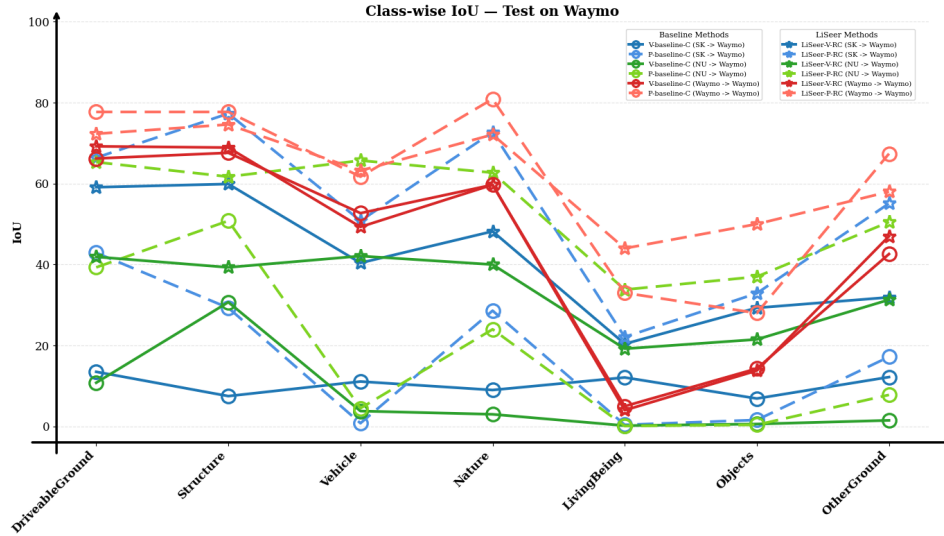
(a) nuScenes (line chart).



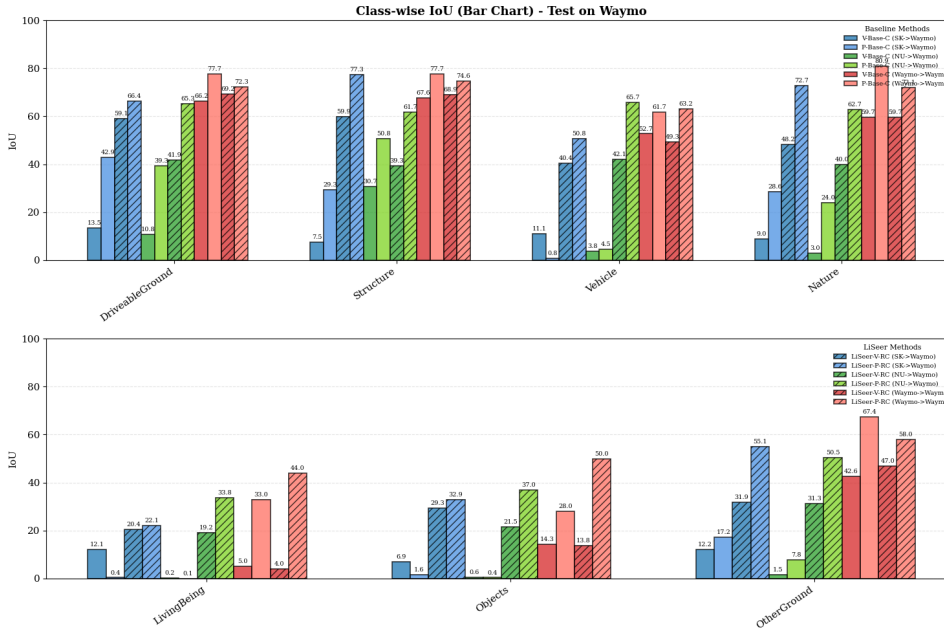
(b) nuScenes (bar chart).

Fig. G10: Class-wise results on nuScenes.

Note. Line and bar charts visualize the same class-wise IoU metrics in different styles.



(a) Waymo (line chart).



(b) Waymo (bar chart).

Fig. G11: Class-wise results on Waymo.

Note. Line and bar charts visualize the same class-wise IoU metrics in different styles.