International Journal of Computer Vision
https://doi.org/10.1007/s11263-024-02166-9

Lidar Panoptic Segmentation in an Open World

Anirudh S. Chakravarthy[1] · Meghana Reddy Ganesina[1] · Peiyun Hu[1] · Laura Leal-Taixé[2] · Shu Kong[3,4] · Deva Ramanan[1] · Aljosa Osep[1]

Received: 15 October 2023 / Accepted: 28 June 2024

Abstract

Addressing Lidar Panoptic Segmentation (*LPS*) is crucial for safe deployment of autnomous vehicles. *LPS* aims to recognize and segment lidar points w.r.t. a pre-defined vocabulary of semantic classes, including `thing` classes of countable objects (e.g., pedestrians and vehicles) and `stuff` classes of amorphous regions (e.g., vegetation and road). Importantly, *LPS* requires segmenting individual `thing` instances (*e.g.*, every single vehicle). Current *LPS* methods make an unrealistic assumption that the semantic class vocabulary is *fixed* in the real open world, but in fact, class ontologies usually evolve over time as robots encounter instances of *novel* classes that are considered to be unknowns w.r.t. thepre-defined class vocabulary. To address this unrealistic assumption, we study *LPS* in the Open World (LiPSOW): we train models on a dataset with a pre-defined semantic class vocabulary and study their generalization to a larger dataset where novel instances of `thing` and `stuff` classes can appear. This experimental setting leads to interesting conclusions. While prior art train class-specific instance segmentation methods and obtain state-of-the-art results on known classes, methods based on class-agnostic bottom-up grouping perform favorably on classes outside of the initial class vocabulary (*i.e.*, unknown classes). Unfortunately, these methods do not perform on-par with fully data-driven methods on known classes. Our work suggests a middle ground: we perform class-agnostic point clustering and over-segment the input cloud in a hierarchical fashion, followed by binary point segment classification, akin to Region Proposal Network (Ren et al. NeurIPS, 2015). We obtain the final point cloud segmentation by computing a cut in the weighted hierarchical tree of point segments, independently of semantic classification. Remarkably, this unified approach leads to strong performance on both known and unknown classes.

Keywords Lidar Panoptic Segmentation · Open world segmentation · Lidar scene understanding

1 Introduction

Lidar Panoptic Segmentation (LPS) (Behley et al., 2021; Fong et al., 2021) unifies lidar point classification and segmentation, both important for autonomous agents to interact with the open environment. In LPS, each point must be clas-

Communicated by Zhun Zhong.

✉ Anirudh S. Chakravarthy
achakrav@andrew.cmu.edu
https://anirudh-chakravarthy.github.io/

Meghana Reddy Ganesina
mganesin@andrew.cmu.edu
https://g-meghana-reddy.github.io/

Peiyun Hu
peiyunh.ph@gmail.com
https://peiyunh.github.io/

Laura Leal-Taixé
leal.taixe@tum.de
https://dvl.in.tum.de/team/lealtaixe/

Shu Kong
skong@um.edu.mo
https://aimerykong.github.io/

Deva Ramanan
deva@andrew.cmu.edu
https://www.cs.cmu.edu/~deva/

Aljosa Osep
aosep@andrew.cmu.edu
https://aljosaosep.github.io/

[1] Robotics Institute, Carnegie Mellon University, Pittsburgh, USA

[2] Dynamic Vision and Learning, TU Munich, Munich, Germany

[3] Faculty of Science and Technology, University of Macau, Zhuhai, China

[4] Department of Computer Science and Engineering, Texas A&M University, College Station, USA

Published online: 19 September 2024

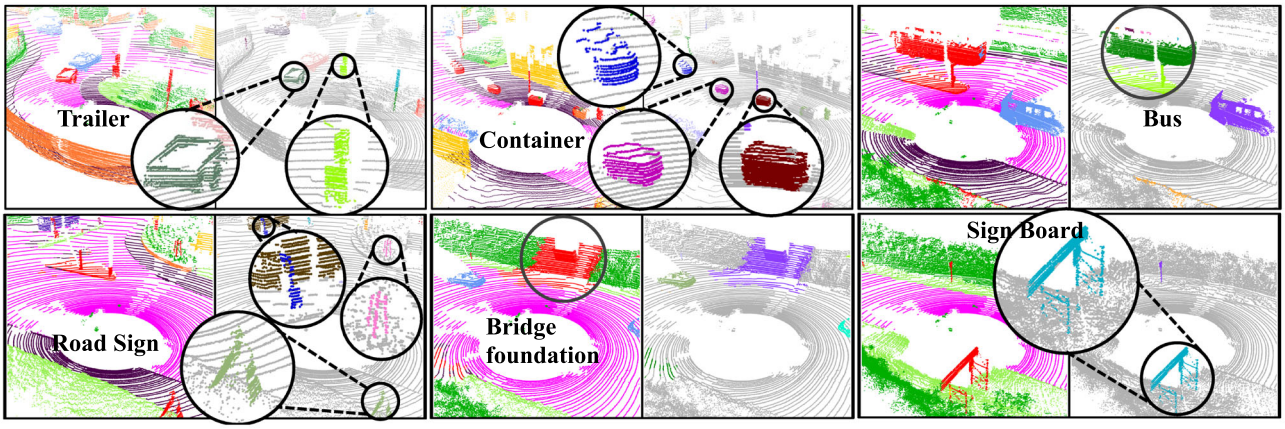


Fig. 1 We study *Lidar Panoptic Segmentation* (*LPS*) in an Open World (*LiPSOW*). In each of the 2×3 subfigure, left panel visualizes segmented points colored different w.r.t semantic classes, where **red** encodes `unknown`; right panel visualizes segmented `thing` instances of `known` and `unknown`. For autonomous navigation, one should evaluate *LPS* methods in the presence of *novel* `thing` object instances and `stuff` regions, which are usually termed as `unknown`. We call this setting *LiPSOW*, where methods should particularly segment points into `unknown` object instances that are outside of the K-way semantic classes in the predefined vocabulary. For example, given the predefined vocabulary by SemanticKITTI (Behley et al., 2019), the `unknown` objects can be trailers, containers, signaling structures, highway bridge foundations, and buses, as visualized in this figure (Color figure online)

sified as one of pre-defined K semantic classes. LPS also defines the notion of `stuff` and `thing`: `thing` classes are countable (*e.g.*, `pedestrian` and `car` classes) and must be assigned unique instance identities. Amorphous regions, such as `vegetation` and `road`, are defined as `stuff`.

1.1 Motivation

Solving *LPS* attracts increasing attention owing to its practical value for robotic applications, but the current setup fails to consider the realistic *open-world* testing environments, where robots must know when they observe regions that do not fit in the predefined vocabulary of K `known` classes (*e.g.*, `fallen-tree-trunk` or `overturned-truck`) and recognize these regions as `unknown` obstacles.

1.2 Lidar Panoptic Segmentation the Open World

We study *LPS in the open-world* (*LiPSOW*, Fig. 1), motivated by real-world challenges: AV companies have already operated autonomous fleets in different geo-locations, and these vehicles constantly observe new or previously `unknown` semantic classes over time. To study such situations, we introduce the LiPSOW evaluation protocol. For example, LiPSOW allows one to train models on the SemanticKITTI (Behley et al., 2019) by using its K common semantic classes for K-way classification of predicted segments, and importantly, gathering all the remaining rare classes as a catch-all `other` class (Hendrycks et al., 2019; Kong & Ramanan, 2021) to better detect `unknown` objects. Further, it performs evaluation on the KITTI360 (Liao et al., 2021) dataset, recorded in the same city with the same sensors but labeling more classes (Lin et al., 2022). This effectively expands the class vocabulary to include instances of `unknown` classes. The main challenge in *LiPSOW* is to recognize and segment K `known` classes as defined in the SemanticKITTI vocabulary, and recognize `unknown` classes that appear in the testing set, i.e., KITTI360.

1.3 Technical Insights

Prior efforts in LPS (Aygün et al., 2021; Gasperini et al., 2021; Hong et al., 2021; Li et al., 2022) *learn* to group `known` classes but fail to generalize to `unknown` classes. Based on this observation, Wong et al. (2020) suggests to learn to segment `known` classes and lean on bottom-up point clustering methods (Teichman et al., 2011; Nunes et al., 2022; Wong et al., 2020; Moosmann et al., 2009) to segment `unknown` instances. Our findings, derived from our publicly available LiPSOW benchmark, suggest unified treatment of `known` and `unknown` classes: (i) we learn which points do *not* correspond to K-known classes via outlier exposure (Hendrycks et al., 2019; Kong & Ramanan, 2021), (ii) segment `unknown` and `thing` classes using class-agnostic bottom-up methods at multiple hierarchies, and (iii) learn which segments in the segmentation tree likely are objects by using labeled data, akin to class-agnostic training of Region Proposal Network (Ren et al., 2015). Surprisingly, this approach not only is

effective for segmenting novel instances with high fidelity but also outperforms learned point-grouping-based methods for known `thing` classes, suggesting a unified treatment of `known` and `unknown` classes.

1.4 Contributions

We make three major contributions: (i) We introduce LiPSOW, a new problem setting that extends *LPS* to the *open-world*, and establish an evaluation protocol to study LiPSOW. (ii) We repurpose existing *LPS* methods to address LiPSOW and comprehensively analyze their performance. (iii), drawing insights from in-depth analysis of existing *LPS* methods, we propose an approach that combines lidar semantic segmentation and a non-learned clustering algorithm with a learned scoring function. Our method effectively segments objects in a class-agnostic fashion, from both `known` and `unknown` classes during testing. To foster future research, we make our code publicly available.

2 Related Work

2.1 Lidar Semantic and Panoptic Segmentation

Recent Lidar Semantic Segmentation (LSS) and Lidar Panoptic Segmentation (LPS) methods are data-driven and fueled by the developments in learning representations from point sets (Qi et al., 2016, 2017; Thomas et al., 2019) and publicly available densely labeled datasets (Behley et al., 2019, 2021; Fong et al., 2021). LSS methods classify points into K classes, for which dense supervision is available during training. Prior works focus on developing strong encoder-decoder-based architectures for sparse 3D data (Qi et al., 2016, 2017; Thomas et al., 2019; Yan et al., 2018; Choy et al., 2019; Tang et al., 2020; Zhu et al., 2021; Loiseau et al., 2022; Ye et al., 2021), the fusion of information obtained from different 3D representations (Alonso et al., 2020; Xu et al., 2021; Li et al., 2022) or neural architecture search (Tang et al., 2020). On the other hand, LPS methods (Behley et al., 2021; Sirohi et al., 2021; Gasperini et al., 2021) must additionally segment instances of `thing` classes. Early methods combine Lidar semantic segmentation networks with 3D object detectors, with a heuristic fusion of both sources of information (Behley et al., 2021). Efficient-LPS (Sirohi et al., 2021) follows two-stage image-based object detection architectures using a range-image-based convolutional backbone. Several methods focus on end-to-end learning using point-based (Thomas et al., 2019; Hong et al., 2021, 2024; Li et al., 2023) or sparse voxel (Zhu et al., 2021) backbones. Recent efforts focus on additional modalities such as camera-lidar fusion (Marcuzzi et al., 2023; Zhang et al., 2023) or different views of Lidar data, such as range-view map (Li et al., 2023) to improve model performance. In addition to learning to classify points, these methods learn to group points in space (Gasperini et al., 2021; Hong et al., 2021; Zhou et al., 2021; Razani et al., 2021; Li et al., 2022), space and time (Aygün et al., 2021; Kreuzberg et al., 2022), or resort to bottom-up geometric clustering to segment instances (Zhao et al., 2022, 2021). Unlike our work, these methods do not consider the open-world environment, in which a pre-fixed class vocabulary is insufficient to capture all semantic classes that are encountered during the autonomous operation.

2.2 Bottom-Up Lidar Instance Segmentation

Bottom-up grouping based on Euclidean distance has been used to isolate object instances in Lidar scans in a class-agnostic manner since the dawn of Lidar-based perception (Thorpe et al., 1991). Existing methods employ techniques such as flood-filling (Douillard et al., 2011; Teichman et al., 2011) and connected components (Klasing et al., 2008), estimated in the rasterized bird's-eye view, bottom-up grouping (Moosmann et al., 2009; Behley et al., 2013; McInnes et al., 2017) using density-based clustering methods (Ester et al., 1996) or graph-based clustering methods (Wang et al., 2012). Nunes et al. (2022) propose to segment object instances with DBSCAN and refine segments using GraphCuts (Boykov & Funka-Lea, 2006). Since one-fits-all clustering parameters are difficult to obtain, Hu et al. (2020) propose constructing a hierarchical tree of several plausible Lidar segmentations, obtained using a density-based clustering method (Ester et al., 1996). These regions are then scored using a learned objectness regressor, and optimal instance segmentation (w.r.t. the regressed objectness function) can then be obtained via a cut in this tree. In this paper, we demonstrate that a data-driven Lidar Panoptic Segmentation network, in conjunction with hierarchical tree construction, forms a strong baseline for Lidar Panoptic Segmentation in an Open World.

2.3 Domain Adaption for Lidar Segmentation

Domain adaptation aims to improve the generalization ability of segmentation models, trained on the source domain, by adapting models to the (unlabeled) target domain. Prior works focus on adapting 3D representations (Langer et al., 2020; Yi et al., 2021), feature representations (Rist et al., 2019; Jiang & Saripalli, 2021; Wu et al., 2019; Shaban et al., 2023; Kong et al., 2023). While our paper focuses on identifying unseen novel objects (`unknown`'s) under similar sensor distributions and geographical regions, these methods focus on adapting to target distributions under significant sensor configuration shifts or environments.

2.4 Open-Set Recognition (OSR)

OSR requires training on data from K known classes and recognizing examples from unknown classes encountered during testing (Scheirer et al., 2012). Many OSR approaches train a K-way classification network and then exploit the trained model for OSR (Yoshihashi et al., 2019; Oza & Patel, 2019). Recent work shows that a more realistic setup is to allow access to some outlier examples during training (Hendrycks et al., 2019; Kong & Ramanan, 2021). Such outliers are used as instances of other class (*i.e.*, held-out samples that do not correspond to pre-defined K-classes) during training, significantly boosting OSR performance. In the context of (lidar) semantic segmentation, Kong and Ramanan (2021); Cen et al. (2022) approximate the distribution of novel objects by synthesizing instances of novel classes. Different from the aforementioned, we tackle OSR through the lens of lidar panoptic segmentation.

2.5 Open-Vocabulary Object Detection

Recent efforts utilize such bottom-up segmentation, combined with Kalman-filter-based object trackers (*c.f.*, (Teichman & Thrun, 2012; Dewan et al., 2015; Osep et al., 2018, 2020)) to pseudo-label instances of moving objects in Lidar Najibi et al. (2022), Zhang et al. (2023) or stereo video streams (Osep et al., 2019, 2018) and use these instances to train object detectors for moving object instances. Moreover, Najibi et al. (2023) demonstrate that detected moving objects can also be classified in a zero-shot manner by distilling CLIP (Radford et al., 2021) features to Lidar. Different from the aforementioned, we tackle *LPS*, which entails dense segmentation and recognition of `thing` and `stuff` classes for moving, as well as stationary objects. Segmented instances that our method classifies as `unknown` class could be further classified in a similar fashion as proposed in Najibi et al. (2023) to obtain a fine-grained semantic interpretation of segmented regions.

2.6 Open-Set (Lidar) Segmentation

Early works by Teichman et al. (2011), Moosmann et al. (2009), Moosmann and Stiller (2013), Held et al. (2016) can be understood as early attempts towards open-set Lidar instance segmentation. In Teichman et al. (2011) after bottom-up segmentation of individual point clouds, objects are tracked across time, and classified as `car`, `cyclist`, `pedestrian` or `other`. The most similar works to ours are Wong et al. (2020), Hwang et al. (2021), which study OSR in lidar point clouds and images, respectively. Their setup assumes complete annotations for `stuff` classes, *i.e.*, assume `stuff` classes exhaustively labeled. This is an unrealistic assumption since new `stuff` classes (*e.g.*, bridges and tunnels) may also be encountered at test time and must be recognized as novel classes. Differently, our setup assumes novel classes (*i.e.*, `unknown`'s classes) can appear in both `stuff` and `thing` classes. This is a realistic setup, further justified by the ontology change from SemanticKITTI to KITTI360 (Lin et al., 2022) where several new `stuff` classes are encountered (*e.g.*, Fig. 3: gate, wall, tunnel, bridge, garage, stop, rail track). This subtle yet crucial distinction separates LiPSOW from previous settings. Prior works (Wong et al., 2020; Hwang et al., 2021; Cen et al., 2022) build their methods and evaluation on the assumption that the unknown consists of only `thing`, *i.e.*, assuming complete annotations for `stuff`. Our experimental validation (Sect. 5.3) confirms that the open-set semantic segmentation method (Cen et al., 2022), which assumes complete annotation for `stuff` classes, does not perform well in our proposed setup. While Wong et al. (2020) is the first work investigating *LPS* in open-set conditions, it conducts experiments on a proprietary dataset, and does not release the code or data. We repurpose publicly available datasets to foster future research on LiPSOW. Finally, we suggest a different approach for LiPSOW that unifies instance segmentation of `known` and `unknown` classes in a class-agnostic manner, by contrast to Wong et al. (2020) that learns to segment `known` classes and *only* segment instances of novel classes via DBSCAN.

3 Open World Lidar Panoptic Segmentation

In this section, we review the problem of Lidar Panoptic Segmentation (*LPS*) and discuss the limitations of its setup from the perspective of open-world deployment in Sect. 3.1. To address the limitations, we introduce *LPS* in an Open World setting (LiPSOW) in Sect. 3.2.

3.1 Lidar Panoptic Segmentation

Definition. *LPS* takes a Lidar point cloud $\mathcal{P} = \{\mathbf{p}_i \in \mathbb{R}^3\}_{i=1}^{N}$ as input, and aims to classify points w.r.t. a predefined vocabulary $\mathcal{K} = \{1, \ldots, K\}$ of K categorical labels and segment object instances. Categorical labels are divided into (1) `thing` classes, covering countable objects such as cars and persons, and (2) `stuff` classes that cover uncountable amorphous regions such as road and vegetation. For `thing` points, *LPS* methods must segment object instances (*e.g.*, every car). Mathematically, *LPS* methods learn a function $f(\cdot; \boldsymbol{\theta})$ parameterized by $\boldsymbol{\theta}$, mapping an input point $\mathbf{p}_i$ to a semantic label k and object instance ID_i, *i.e.*, $f(\mathbf{p}_i; \boldsymbol{\theta}) \rightarrow (k, \mathrm{ID}_i)$, where $k \in \mathcal{K}$, ID_i is a unique ID for the object instance $\mathbf{p}_i$ belongs to, and particularly, $\mathrm{ID}_i = 0$ means that $\mathbf{p}_i$ belongs to one of the `stuff` classes (*i.e.*, class-k is `stuff`). *LPS* measures the per-point classification accuracy

(*i.e.*, Lidar semantic segmentation) and per-instance segmentation accuracy.

Remarks. *LPS* does not properly formulate the real-world case that there exist points belonging to an `unknown` catch-all superclass, which contain various unknown classes encountered only during testing. For example, a vocabulary in AVs probably does not have labels such as `sliding-unattended-stroller` and `fallen-tree-trunk`, but AVs must segment them into individual instances for safe maneuver such as "stop", "yield" or "change-lane". We are motivated to address this in detail below.

3.2 LPS in an Open World

Definition. Extending *LPS*, *Lidar Panoptic Segmentation in the Open World* (LiPSOW) further requires classifying points into an `unknown` class if the points do not belong to any of the predefined K semantic classes, and segment them as `unknown` instances. That said, `unknown` covers all classes that do not correspond to any of the K predefined classes and might contain unannotated instances that, without prior knowledge, cannot be treated as a `stuff` or `thing` class. Formally, we define the `unknown` class as the $(K+1)^{th}$ class, so LiPSOW methods learn a function $f(\cdot;\boldsymbol{\theta})$ parameterized by $\boldsymbol{\theta}$, mapping input points $\mathbf{p}_i$ to a semantic label k and instance ID_i, *i.e.*, $f(\mathbf{p}_i;\boldsymbol{\theta}) \rightarrow (k, \mathrm{ID}_i)$, where $k \in \{1,\ldots,K,K+1\}$. As before, ID_i is a unique ID for point-i and $\mathrm{ID}_i = 0$ implies that class-k is `stuff`.

3.2.1 Significance

By definition, LiPSOW algorithms should be able to distinguish `unknown` objects from the predefined K classes and to segment corresponding object instances. This ability is useful for many applications. First, recognizing unknown objects is crucial for safety-critical robotic operations, *e.g.*, AVs should recognize a never-before-seen `sliding-unattended-stroller` to avoid collision and casualty. Second, unknown instances could be used in conjunction with active learning (Ren et al., 2021; Zhan et al., 2022) to help select more valuable examples that are recognized as unknown to reduce data collection and annotation costs.

Remark I. **Can `unknowns` be seen during training?** Related to LiPSOW is open-set recognition, the task of recognizing `unknown` examples during test-time. A conventional setup is that all the training data is labeled w.r.t. the K predefined classes, and test-time examples may originate from any of the K `known` classes, or the $(K+1)^{th}$ `unknown` (Scheirer et al., 2012; Bendale & Boult, 2016; Yoshihashi et al., 2019; Oza & Patel, 2019) class. While these works suggest that `unknown` examples should not be part of the training set, recent work has comprehensively demonstrated that a more reasonable setup is to explicitly exploit outlier data or diverse `other` examples during training. In particular, Hendrycks et al. (2019); Kong and Ramanan (2021) show that such models effectively generalize to real unseen examples. To improve the real-world AV application, we consider the latter setup, *i.e.*, the training set contains instances of `known` classes, labeled as `other`. Instances of these classes are available during the training, but, importantly, do not overlap with the K `known` classes. These classes are presented as possible instances of the `unknown`, $(K+1)^{th}$ class. Note that the `other` class is often named `void` or `unlabeled` in many contemporary benchmarks (Cordts et al., 2016; Neuhold et al., 2017).

Remark II. **How to define `unknown` semantic classes?** Strictly defining all objects that could appear in the open-world would be difficult. Instead, inspired by the image segmentation literature (Martin et al., 2001; Fomenko et al., 2022), which suggests that humans have a consensus on which image regions constitute object instances, we define `unknown` instances as those that were (i) not annotated w.r.t. a categorical label in the train-set but (ii) labeled as objects by human annotators in the test-set.

Remark III Can only novel `thing` classes be regarded as `unknown`'s? Prior efforts tackling open-set in images and point clouds (Wong et al., 2020; Hwang et al., 2021) assume `stuff` classes exhaustively labeled. This is an unrealistic assumption since new `stuff` classes (*e.g.*, Fig. 3: tunnel, bridge, rail track, etc.) may also be encountered at test time and must be recognized as novel classes. This subtle yet crucial distinction separates LiPSOW from prior work.

4 Methodology

Existing methods for *LPS* (Aygün et al., 2021; Gasperini et al., 2021; Zhou et al., 2021; Razani et al., 2021) learn to classify points, and *learn* to group points that represent `thing` classes. These methods work under the *LPS* setting, where semantic and instance-level supervisions are given for all classes. For LiPSOW, we need to rethink existing methodology: LiPSOW methods must (just as in *LPS*) recognize predefined semantic classes and segment instances of `thing` classes. Additionally, they need to cope with the inherent difficulty of recognizing and segmenting the `unknown` class that *mixes* `stuff` and `things`. Unfortunately, for this catch-all class, exhaustive semantic and instance-level supervision is not available.

To design a LiPSOW method, named *Open-World Lidar Panoptic Segmentor* (OWL), we draw inspiration from two-stage object detectors (Ren et al., 2015). It was shown in literature that such networks can be repurposed for image-based open-set (Dhamija et al., 2018) and open-world object detection (Hwang et al., 2021; Weng et al., 2021; Liu et al., 2022; Joseph et al., 2021; Fomenko et al., 2022). Two-

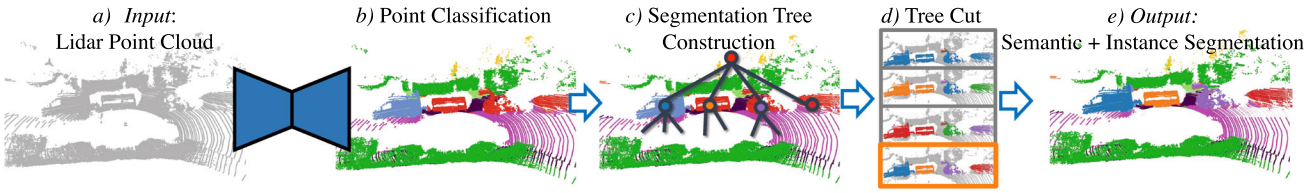


Fig. 2 **Open-World Lidar Panoptic Segmentation (OWL)**: We first perform $K+1$ semantic segmentation network on a point cloud (**a**, **b**) and classify points as `stuff`, `things`, and **`unknown`** (point color encodes semantic classes, **red** points represent `unknown`'s). Then we construct a hierarchical tree of "all possible" segments for `thing` and **`unknown`** points (**c**) and train a segment-scoring function to cut the tree (**d**), finally producing instance and semantic segmentation results (**e**) (Color figure online)

stage networks were also adopted for 3D object detection (Shi et al., 2019; Chen et al., 2015). However, a 3D analogy of the region proposal network (RPN), a key component that allows us to recognize instances of novel classes, is not trivial due to the large 3D search space (Shi et al., 2019; Chen et al., 2015). For this reason, prior works constrain the set of anchor boxes to the mean size of each semantic class (*e.g.*, `car` and `pedestrian` sized boxes). This approach is not scalable to the large variety of object instances that may appear in the `other` class. Instead, we rely on the observations of early work on Lidar perception (Teichman et al., 2011; Teichman & Thrun, 2012; Held et al., 2016; Behley et al., 2013), which shows that simple bottom-up grouping of points yields a compact set of class-agnostic object candidates. In the following sections, we outline a simple and effective method for LiPSOW, based on data-driven approaches (Thomas et al., 2019; Aygün et al., 2021) and perceptual grouping (Klasing et al., 2008; Douillard et al., 2011; Hu et al., 2020; Behley et al., 2013) based methods, as well as recent findings in open-set recognition (Kong & Ramanan, 2021).

4.1 High-Level Overview

We propose a two-stage network for LiPSOW, which is trained sequentially. We adopt an encoder-decoder point-based backbone (Thomas et al., 2019) to learn to classify points in $K+1$ fashion. We explicitly train our network to distinguish points from K `known` classes from the `other` class, that is considered to be a representative of the `unknown` class during the model training (Fig. 2b). This network estimates label predictions that belong to `stuff`, `things`, and the mixed `other` class. In the second stage, we run a non-learned clustering algorithm on points recognized as `thing` or `other` (Fig. 2c) and apply a learned scoring function to derive the final instance segmentation. To this end, we produce a hierarchical segmentation tree (*c.f.*, Hu et al. (2020)), and train the second-stage network that learns to estimate how likely a point segment is an object, and run a min-cut algorithm Hu et al. (2020) to obtain a unique, globally optimal point-to-instance assignment (Fig. 2d). Importantly, this method treats the `thing` and `other` classes in a unified manner, producing instance segmentation for both. We present individual components of our *Open-World Lidar Panoptic Segmentor* (OWL) baseline below.

4.2 Semantic Segmentation Network

We train an encoder-decoder architecture that operates directly on point cloud $P \in \mathbb{R}^{N\times 3}$. In particular, we train a well-consolidated KPConv (Thomas et al., 2019) network with deformable convolutions; however, we note that a variety of backbones suitable for learning representations from unstructured point sets could be used (Qi et al., 2016; Yan et al., 2018). We use the KPConv-based LPS network (Aygün et al., 2021) due to its (i) open-source implementation and (ii) point-based backbone, that directly learns fine-grained per-point features, as opposed to 3D sparse-convolutional networks (Choy et al., 2019) that estimate per-voxel features. We attach a semantic classifier on top of the decoder feature representation $F \in \mathbb{R}^{N\times D}$ that outputs a semantic map $S \in \mathbb{R}^{N\times(K+1)}$. Finally, we train this network head using the cross-entropy loss. The difference from conventional (lidar) semantic segmentation training is that we explicitly introduce an additional catch-all class by holding out rare (`other`) classes during the model training (Sect 5.1.2). This class is analogous to the catch-all *background* class (Ren et al., 2015), a common practice in training two-stage object detectors. The final $(K+1)$-way softmax provides a smooth distribution that indicates the likelihood of a point being one of K classes or the `other` class. Classes classified as `other` during test time are considered to be `unknown`'s.

Such a catch-all `other` class is not common practice in training semantic segmentation networks, as it is usually assumed that points (or pixels) are densely and exhaustively labeled. However, in LiPSOW, this assumption is no longer valid. Without it, we would incentivize the network to label each point as one of the K classes.

4.3 Segmenting any Object

Using a proximity-based point grouping method, we can construct combinatorially many possible point segments from a point cloud of size N. We learn a function $f(p) \rightarrow [0, 1],\ p \subset P \in \mathbb{R}^{N\times 3}$ that scores *objectness* of a subset of points in a data-driven manner to indicate *how likely* a point segment encapsulates an object. This is analogous to image-based object proposal generation methods (Alexe et al., 2012; Zitnick & Dollár, 2014) that adopt sliding window search and learn an objectness score to rank windows. The advantage of such methods over recent work on data-driven pixel/point grouping (Kong & Fowlkes, 2018; Aygün et al., 2021; Razani et al., 2021; Zhou et al., 2021) is that the set of possible objects should naturally cover all objects, irrespective of class labels.

To understand if our segmentation tree covers most of the relevant objects, we measure recall using labeled instances. To this end, we follow (Hu et al., 2020) and construct a hierarchical segmentation tree T by applying Euclidean clustering recursively with decreasing distance threshold using DBSCAN (Ester et al., 1996) and parameters recommended by Hu et al. (2020). Our experiments show that with this approach we can recall 97.2% of instances labeled in the SemanticKITTI dataset (Behley et al., 2019, 2021) validation set (see Sect. 5.2). This shows that not only can this approach segment a large variety of objects, but also it does not need to learn how to group instances of `known` classes. *These instances are already included in the segmentation tree.*

4.4 Learning an Objectness Function

There are several ways to learn such a function $f(p) \rightarrow [0, 1]$ that estimates how likely a subset of points represents an object. One approach is to estimate a per-point objectness score. Following (Aygün et al., 2021), this can be learned by regressing a truncated distance $O \in \mathbb{R}^{N\times 1}$ to the nearest point center of a labeled instance (Aygün et al., 2021) atop of decoder features $F \in \mathbb{R}^{N\times D}$. The objectness value can then be averaged over the segment $p \subset P$. Alternatively, we can train a holistic classifier as a second-stage network by pooling point segment features, followed by fully-connected layers, similar to the PointNet (Qi et al., 2016) classification network. In this case, we pre-built hierarchical segmentation trees T_i for each point cloud i in the training set and minimize the training loss based on the signal we obtain from *matched* segments between the segmentation trees and set of labeled instances, GT_i. One possibility is to use binary cross-entropy loss (similar to how the RPN is trained); alternatively, we can directly regress the objectness value to be proportional to the point-set intersection-over-union. We detail the network architecture and training recipes in the appendix ("B Implementation Details Section") and discuss design choices on how to train such a network in Sect. 5.2.

4.5 Unique Point-to-Instance Assignment

Segmentation tree T provides a hierarchy of pairs of point segments and their corresponding scores. However, for LiPSOW, we need to assign points to instances uniquely. Intuitively, this property will be satisfied with any "cut" in this tree; we could simply output leaf nodes in a tree after the cut is performed. The question then boils down to *where* to cut such that the overall segmentation score is as good as possible according to some criterion. It was shown in Hu et al. (2020) that we can compute optimal worst-case segmentation efficiently by simply traversing the tree, as long as we have *strictly smaller* segments at each tree-level. Optimal worse-case segmentation is the segmentation that yields an overall (global) segmentation score when the overall segmentation score is defined as the *worst* objectness among its local segments (this can be efficiently evaluated by looking at the tree leaf nodes). This approach ensures a unique point-to-segment assignment *i.e.* no overlap. This algorithm is not the contribution of this paper. However, for completeness, we provide the algorithm with a detailed explanation in the appendix ("C.1 Segmentation Tree Generation" Section).

4.6 Inference

At inference time, we first make a forward pass through our network, construct the segmentation tree on points classified as `thing` or `other` (*i.e.* `unknown`) class, and run our objectness classifier for each segment in the tree. After such construction, we run the tree-cut algorithm to obtain unique point-to-instance assignments. Finally, as semantic labels within segments may be inconsistent, we assign a majority vote to each segmented instance.

5 Experiments

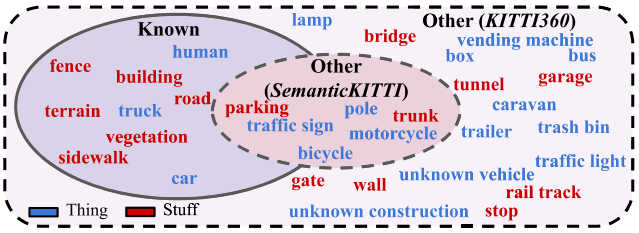
In this section, we first outline our experimental setup for LiPSOW (Sect. 5.1). Then, we discuss and ablate our *Open-World Lidar Panoptic Segmentor* (OWL) (Sect. 5.2) on a standard lidar panoptic segmentation (*LPS*) benchmark, SemanticKITTI (Behley et al., 2019, 2021). Finally, we demonstrate the generality of our approach with cross-dataset evaluation (Sect. 5.3).

5.1 Evaluation Protocol of LiPSOW

We set up an evaluation protocol to simulate the conditions that occur when a robot within a certain geographic region (*i.e.*, city), *e.g.*, a robot taxi fleet. Here, data is recorded by the

	SemanticKITTI	KITTI360
Sensor	Velodyne HDL-64E	Velodyne HDL-64E
Geographic region	Karlsruhe, GER	Karlsruhe, GER
Distance	39.2 km	73.7 km
Recording date	2011	2013
# Scans	23k	80k
# Classes	21	37
# Instance classes	8	24

(a)



(b)

Fig. 3 We base LiPSOW setup on SemanticKITTI (Behley et al., 2019, 2021) and KITTI360 (Liao et al., 2021) datasets. We train and validate models using SemanticKITTI, and re-purpose KITTI360 dataset, which contains classes, not labeled in SemanticKITTI (*i.e.*, `unknown`) as a test set. A detailed list of our taxonomy is provided in the appendix (Table 5) (Color figure online)

same sensor type (cross-sensor domain generalization is out-of-scope of this paper), which is reasonable in practice when deploying robot taxis of the same type. Importantly, however, even though we focus on a certain geographic region, *source* and *target* data must be recorded at disjunct locations (*i.e.*, sequences that appear in the test set should not be recorded in precisely same city districts). In such a setting, domain shifts are often gradual *e.g.*, we record more data over time, and therefore observe a larger variety of `known` regions and more `other` objects that appear in the long-tail of the object class distribution.

5.1.1 Open-World Lidar Panoptic KITTI

To study LiPSOW in such a setting, we base our experimental setup on SemanticKITTI (Behley et al., 2019) and KITTI360 (Liao et al., 2021) datasets. They were recorded in *distinct* regions of Karlsruhe, Germany (Table 3a). We use SemanticKITTI for model training and validation, and KITTI360 sequences *only* for testing.

5.1.2 Source (Train/Val) Domain

In Fig. 3b (blue "known" set) we visualize the SemanticKITTI classes (Behley et al., 2019). The dashed inner circle (pink) denotes the rarer classes, which we merge into a single `other` class. Those are examples of regions, different from K known classes, *i.e.*, $(K+1)^{th}$ catch-all class. This allows evaluating how well the model learns to separate `known` classes from `other` by measuring IoU for the `other` class and mIoU for all classes within the source domain. We call this taxonomy Vocabulary 1. However, by this construction of Vocabulary 1, classes such as `bicycle` and `motorcycle`, which belong to `other`, are important for autonomous driving and commonly observed in urban environments. Therefore, we also construct Vocabulary 2, which holds out only the rarest categories as `other`. We provide further details on vocabulary construction and taxonomy in the appendix ("A LiDAR Panoptic Segmentation in Open-World" Section and Table 5).

5.1.3 Target (Test) Domain

We evaluate models on KITTI360, which encompasses all SemanticKITTI classes, and *importantly*, additional 10 `thing` (with instance labels) and 7 `stuff` classes, which are used as novel classes in experiments. We discuss vocabulary changes that ensure SemanticKITTI and KITTI360 vocabularies are consistent in the appendix ("A LiDAR Panoptic Segmentation in Open-World" Section and Table 5).

5.1.4 Metrics

We repurpose evaluation metrics proposed in the context of semantic segmentation (mean intersection-over-union, mIoU (Everingham et al., 2010)) and panoptic segmentation (panoptic quality, PQ (Kirillov et al., 2019)). To quantify the point classification performance (mIoU), we simply treat `other` class as "just one more class". To quantify panoptic segmentation, we split the evaluation for `known` classes and `other` classes. We evaluate `known` using $PQ = SQ \times RQ$, as defined by Kirillov et al. (2019). The segmentation (SQ) term averages instance-level IoU for each true positive (TP), while the recognition quality $\left(RQ = \frac{\text{TP}_c}{|\text{TP}_c| + \frac{1}{2}|\text{FP}_c| + \frac{1}{2}|\text{FN}_c|}\right)$ is evaluated as F1 score (harmonic mean of precision and recall). For `other` classes, the task definition does not specify which semantic classes are `thing` classes, nor specifies the vocabulary of target instance classes. As we cannot annotate every possible semantic class, it is important to not penalize *false positives*, FPs, as these cannot be clearly defined. Therefore, we follow Wong et al. (2020); Liu et al. (2022), and replace the RQ term with recall $\left(\frac{\text{TP}_c}{|\text{TP}_c| + |\text{FN}_c|}\right)$, which we call UQ (unknown quality). Note that our UQ is computed slightly differently from the UQ introduced by Wong et al. (2020) because we do not penalize segment

Table 1 LPS results on SemanticKITTI validation set.

	Method	PQ	PQ†	RQ	SQ	PQ^{Th}	RQ^{Th}	SQ^{Th}	PQ^{St}	RQ^{St}	SQ^{St}	mIoU	$Prec^{Th}$	Rec^{Th}
Baselines	Panoster (Gasperini et al., 2021)	55.6	–	66.8	79.9	56.6	65.8	–	–	–	–	61.1	–	–
	DS-Net (Hong et al., 2021)	57.7	63.4	68.0	77.6	61.8	68.8	78.2	54.8	67.3	77.1	63.5	–	–
	DS-Net v2 (Hong et al., 2024)	61.4	65.2	72.7	79.0	65.2	72.3	79.3	57.9	71.1	79.3	69.6	–	–
	PolarSeg-Panoptic (Zhou et al., 2021)	59.1	64.1	70.2	78.3	65.7	74.7	87.4	54.3	66.9	71.6	64.5	–	–
	MaskPLS (Marcuzzi et al., 2023)	59.8	–	69.0	76.3	–	–	–	–	–	–	61.9	–	–
	Efficient-LPS (Sirohi et al., 2021)	59.2	65.1	69.8	75.0	58.0	68.2	78.0	**60.9**	71.0	72.8	64.9	–	–
	GP-S3Net (Razani et al., 2021)	**63.3**	**71.5**	**75.9**	**81.4**	**70.2**	**80.1**	86.2	58.3	**72.9**	**77.9**	**73.0**	–	–
	Location Guided (Xian et al., 2022)	59.0	63.1	69.4	78.7	65.3	73.5	**88.5**	53.9	66.4	71.6	61.4	–	–
	CPGNet (Li et al., 2022)	62.7	67.5	–	–	70.0	–	–	57.3	**–**	**–**	67.4	–	–
	Panoptic-PHNet (Li et al., 2022)	61.7	–	–	–	69.3	–	–	–	–	–	65.7	–	–
	LCPS (Zhang et al., 2023)	59.0	68.8	68.9	79.8	–	–	–	–	–	–	63.2	–	–
	4DPLS (Aygün et al., 2021)	56.5	61.9	66.8	79.0	57.3	64.3	88.0	56.0	68.6	72.4	64.8	63.3	75.3
Ablations	OWL (class-specific)	58.7	64.1	68.8	79.8	62.2	68.7	89.8	56.1	68.8	72.5	65.0	79.7	69.3
	OWL (class-agnostic)	58.7	64.1	68.8	79.8	62.3	68.8	89.8	56.1	68.8	72.5	65.0	79.9	69.3
	OWL (+ holistic classifier)	58.8	64.2	68.9	79.8	62.5	68.9	89.8	56.1	68.8	72.5	65.0	80.0	69.6
	OWL (+ regression loss)	58.9	64.3	69.0	79.8	62.8	69.2	89.9	56.1	68.8	72.5	65.0	80.0	70.0
	OWL (+ majority vote)	59.0	64.4	68.9	80.0	63.0	69.1	90.3	56.1	68.8	72.5	64.2	81.9	67.4
Oracle	OWL (obj. oracle)	59.2	64.6	69.2	80.0	63.5	69.8	90.2	56.1	68.8	72.5	65.0	80.3	70.9
	OWL (+ majority vote)	59.7	65.1	69.4	80.6	64.6	70.2	91.8	56.1	68.8	72.5	64.2	82.3	69.1
	OWL (GT semantic map)	98.3	98.3	99.2	99.0	96.1	98.3	97.8	100.0	100.0	100.0	100.0	99.4	97.2

Bold values indicate the best result

Alhough LPS is not our primary focus, we show “closed-world" panoptic accuracy, since good performance on `known` classes is a pre-requisite for LiPSOW

Our method is competitive with state-of-the-art methods that adopt stronger semantic backbones. Given GT semantic labels, our method achieves nearly perfect PQ (98.3%), indicating that future efforts should focus on semantic classification

predictions that overlap unlabeled `stuff` in the `other` class.

5.2 Lidar Panoptic Segmentation

A pre-requisite for good performance on LIPSOW is good performance on `known` classes. Therefore, first, we present a comparison of our method with state-of-the-art *LPS* methods on SemanticKITTI Behley et al. (2019) in Table 1. The top performing method on the val-set for `known` classes is *GP-S3Net* Razani et al. (2021). Due to its strong Transformer-based backbone, *GP-S3Net* obtains 73.0% mIoU and the highest PQ of 63.3%. The remaining methods are in the ballpark of 60–65% mIoU and 55–59% PQ. We note that methods such as *CPGNet* (Li et al., 2022) and *LCPS* (Zhang et al., 2023) are also highly performant, however, these utilize multi-modal inputs (range-views and images, respectively). We base our OWL on well-consolidated and easily-extendable KPConv (Thomas et al., 2019), similar to 4D-PLS.[1]

5.2.1 Semantic Oracle

With the semantic oracle experiment, we aim to answer the question: *how far we can get in LPS with our baseline*? To answer this, we replace our learned classification network with ground-truth semantic maps (*GT semantic map*), available in the validation set, but retain our instance branch. This yields a near-perfect PQ of 98.3%. As evident, we can recall 97.2% of `thing` objects with 99.4% precision. This experiment raises questions of whether point-to-instance grouping needs to be learned; *nearly all labeled instances are already included in our segmentation tree*. Moreover, this experiment suggests that future efforts should focus on driving further the *point-classification* performance and that *semantic labeling may be sufficient* – near-perfect instance labels can be obtained via hierarchical clustering.

5.2.2 Objectness Oracle

Given a KPConv-based semantic network, *what performance can we obtain with perfect objectness scoring function?* To answer this, we use GT instance labels to score all segments as IoU between a segment and its best-matching GT segment (*obj. oracle*). With this approach, we obtain 59.2% PQ. By propagating semantic *majority vote* within each segment, we can further improve precision and recall and, consequently, PQ (+0.5%). This is an upper bound that we can obtain with the KPConv semantic backbone.

[1] GP-S3Net (Razani et al., 2021), DSNetv2 (Hong et al., 2024), Panoptic-PHNet (Li et al., 2022) do not provide source code.

5.2.3 Ablations

We start with a variant of our network with semantic and objectness heads (*c.f.*, (Aygün et al., 2021)). In this case, we compute per-segment objectness by averaging per-point objectness scores (see Sect. 4). The *class-specific* variant builds the segmentation tree separately for each semantic class, while *class-agnostic* variant builds it upon all `thing` and `other` points, effectively dropping fine-grained semantic information. With both, we obtain a PQ of 58.7%, +2.2% improvement over 4D-PLS, which uses identical segmentation and objectness networks for inference. In the class-agnostic variant, we observe improvement of +0.2% in terms of precision (79.9%). We conclude this approach is more accurate because it is less sensitive to errors in semantic classification (*e.g.*, a truck, part classified as truck and part as car, cannot be holistically segmented with *class-specific* variant). That said, we can safely treat `thing` and `other` classes in a unified manner. Next, instead of averaging per-point objectness, we train a holistic second-stage objectness classifier using cross-entropy loss (see Sect. 4) and observe PQ improvement (58.8%, +0.1%). This way we gain an additional +0.1% in PQ. We hypothesize that by training our network using regression loss, we obtain smoother objectness scores compared to sharp-peaked (and overconfident) binary classification scores, which is beneficial for the tree-cut algorithm.

5.3 Open-World Lidar Panoptic Segmentation

We now study *within*-dataset performance on SemanticKITTI and *cross*-dataset performance on SemanticKITTI → KITTI360 for two different source-domain vocabularies in Tab 3. *Vocabulary 1* merges rarer classes into a catch-all `other` class (as discussed in Sect. 5.1) while *Vocabulary 2* closely follows the original SemanticKITTI class definitions. We provide further details on the construction of these vocabularies in Appendix "A LiDAR Panoptic Segmentation in Open-World" Section. We report results of known `thing` and `stuff` classes using Panoptic Quality and mean-IoU, and for `unknown` class (*i.e.* points classified as `other` during inference on KITTI360), we report Unknown Quality (UQ), Recall, and IoU. We analyze open-world generalization based on *cross*-dataset performance (*i.e.* `unknown`) and not on *within*-dataset performance.

5.3.1 Baselines

We compare our OWL to vanilla 4D-PLS (Aygün et al., 2021), trained in a single-scan setting. This network uses the same KPConv backbone (Thomas et al., 2019). We also train PolarSeg-Panoptic (Zhou et al., 2021), one of

Table 2 Results of Lidar Semantic Segmentation. Methods are trained on SemanticKITTI under the specified vocabulary, and evaluated on the SemanticKITTI validation set. Experiments indicate that OSeg (Cen et al., 2022)) struggles to generalize to held-out `other` classes

Vocabulary	Method	`known` mIoU	`other` IoU
Vocabulary 1	4DPLS (Aygün et al., 2021)	79.8	56.9
	PolarSeg-Panoptic (Zhou et al., 2021)	74.2	47.3
	OSeg (Cen et al., 2022)	67.3	1.5
Vocabulary 2	4DPLS (Aygün et al., 2021)	70.5	50.8
	PolarSeg-Panoptic (Zhou et al., 2021)	63.7	40.0
	OSeg (Cen et al., 2022)	57.6	0.5

the top-performers on standard *LPS* (see Table 3.1) for which *source code is available*. 4D-PLS† modifies the inference procedure: for points classified as `other`, we lower the center-objectness threshold. We provide details in the appendix ("B Implementation Details" Section). This is based on the intuition that the objectness head of 4D-PLS should be able to generalize to novel classes, but with lower confidence. Finally, OWL ‡ is a modified variant of OWL that uses a learned point-grouping mechanism (*c.f.*, Aygün et al. (2021)) for `thing` classes, and hierarchical segmentor for the `other` class (*i.e.*, does not treat instance segmentation of `thing` and `other` classes in a unified manner). This baseline is inspired by Wong et al. (2020). However, we use 4D-PLS as a backbone network and bottom-up grouping as described in Sect. 4.

5.3.2 Lidar Semantic Segmentation

We compare our approach to OSeg (Cen et al., 2022), a state-of-the-art method for open-set Lidar Semantic Segmentation (LSS) in Table 2. OSeg *only* tackles the semantic point classification of Lidar scans and does not address the instance segmentation aspect of Lidar Panoptic Segmentation. OSeg consists of two stages: (i) Open-set semantic segmentation, where each point is classified into one of K known or a catch-all class using redundant classifiers, and (ii) Incremental learning, where the catch-all categories are incorporated into the model. For a fair comparison with our setting, we report the performance of the first stage of OSeg, without performing incremental learning. We validate performance using our proposed vocabulary splits on SemanticKITTI using publicly available implementation (details in the appendix). As OSeg performs only point classification (*i.e.*, semantic segmentation), we can compare to OSeg only in terms of mean intersection-over-union with our base network for point classification (4DPLS). Results show that OSeg (Cen et al., 2022) significantly underperforms compared to our base 4DPLS network on both `known` and `unknown`. OSeg performance on the `unknown` class performances may be caused by the object synthesis, which implicitly assumes the other category consists of only `thing` and not `stuff` classes, violating the spirit of LiPSOW. As can be seen, our 4DPLS-based network additionally outperforms OSeg on `known` classes.

5.3.3 Lidar Panoptic Segmentation

Vocabulary 1 In Table 3, OWL is top-performer for `known` classes (+1.6% w.r.t. 4D-PLS and +0.8% w.r.t. PolarSeg-Panoptic). Similarly for `other`, OWL recalls 48.4% of objects, compared to 10.8% recalled by 4D-PLS and 14.7% by PolarSeg-Panoptic. Note that the only `other` objects with instance labels in SemanticKITTI are `bicycle` and `motorcycle`. For the `unknown` class in the cross-domain section (KITTI360), we observe 4D-PLS recalls only 2.0% of instances (1.3% UQ). By changing the inference, 4D-PLS† recalls 6.0% of objects. OWL recalls 45.1% of objects, leading to 36.3% UQ. This suggests there is a significant potential to improve further without modifying our instance segmentor: the bottleneck seems to be the point-level classifier (see low IoU of 11.4%). This result also highlights that SemanticKITTI by itself is not sufficient for studying open-world *LPS* methods due to a limited number of classes with instance labels. In KITTI360 we have a significantly larger number of instances in the `unknown` class, exposing the poor generalization of methods trained to segment only the K-known classes.

In SemanticKITTI→ KITTI360, we observe a performance drop in cross-domain evaluation, including `known` classes. 4D-PLS performance drops from 67.8% (SemanticKITTI) to 56.1% PQ (KITTI360) and 79.8% → 65.3% mIoU. PolarSeg-Panoptic works well when evaluated in a within-domain setting but fails to generalize to KITTI360 (0.7% PQ). This suggests that existing models are very sensitive to data distribution shifts. Future efforts should aim not only to improve point classification performance within-domain but also in cross-domain settings.

Vocabulary 2 This setting follows more closely the official SemanticKITTI vocabulary and exposes a smaller number of semantic classes as instances of the `other` class during training. We observe that *Vocabulary 1* generalizes much better across datasets, suggesting that grouping rare classes in a catch-all `other` class leads to better generalization.

Table 3 Open-World Lidar Panoptic KITTI

		Method	known								unknown			
			PQ	RQ	SQ	PQ^{Th}	PQ^{St}	mIoU	$Precision^{Th}$	$Recall^{Th}$	UQ	Recall	SQ	IoU
Vocabulary 1	SemKITTI	4D-PLS (Aygün et al., 2021)	67.8	78.5	85.4	60.0	71.7	79.8	66.1	85.4	7.8	10.8	71.9	56.9
		4D-PLS†	67.6	78.3	85.4	59.4	71.7	79.8	65.5	85.1	19.9	27.6	72.0	56.9
		PolarSeg-Panoptic (Zhou et al., 2021)	68.6	**80.2**	83.6	**68.2**	68.8	74.2	**79.5**	76.8	10.2	14.7	69.3	47.3
		OWL‡	67.8	78.5	85.4	60.0	71.7	79.8	66.1	85.4	**39.8**	**48.5**	**82.1**	56.9
		OWL (Ours)	**69.4**	79.5	**86.3**	64.7	**71.7**	**79.8**	69.3	**87.6**	39.6	48.4	81.8	**56.9**
	KITTI360	4D-PLS (Aygün et al., 2021)	56.1	67.4	80.5	56.2	56.0	65.3	63.5	70.8	1.3	2.0	65.7	11.4
		4D-PLS†	55.8	67.2	80.4	55.4	56.0	65.3	62.5	70.5	4.2	6.0	70.6	11.4
		PolarSeg-Panoptic (Zhou et al., 2021)	0.7	0.9	73.0	0.7	0.7	1.6	14.6	0.6	0.0	0.1	76.4	9.0
		OWL‡	56.1	67.4	80.5	56.2	56.0	65.3	63.5	70.8	**36.6**	**45.4**	**80.6**	11.4
		OWL (Ours)	**59.4**	**70.3**	**81.8**	**66.2**	**56.0**	**65.3**	**72.5**	**78.5**	36.3	45.1	80.5	**11.4**
Vocabulary 2	SemKITTI	4D-PLS (Aygün et al., 2021)	60.2	69.3	81.4	57.9	61.4	70.5	70.2	72.9	16.4	22.2	73.8	50.8
		4D-PLS†	60.1	72.7	81.4	57.6	61.4	70.5	69.9	72.8	20.7	29.0	71.3	50.8
		PolarSeg-Panoptic (Zhou et al., 2021)	58.6	68.2	79.4	56.5	55.2	63.7	72.9	63.7	14.9	20.7	72.1	40.0
		OWL‡	60.2	69.9	81.4	57.9	61.4	70.5	70.2	72.9	49.2	56.9	**86.5**	50.8
		OWL (Ours)	**61.9**	**73.9**	**82.3**	**62.9**	**61.4**	**70.5**	**74.3**	**75.2**	**49.3**	**57.0**	86.3	**50.8**
	KITTI360	4D-PLS (Aygün et al., 2021)	45.9	56.8	77.5	47.9	44.8	53.4	57.3	59.1	3.4	4.9	69.8	6.3
		4D-PLS†	45.8	56.7	77.5	47.7	44.8	53.4	56.9	59.1	5.5	7.9	69.6	6.3
		PolarSeg-Panoptic (Zhou et al., 2021)	1.2	1.8	61.7	0.4	1.6	3.0	9.2	0.3	0.0	0.0	71.1	0.7
		OWL‡	45.9	56.8	77.5	47.9	44.8	53.4	57.3	59.1	**21.3**	**26.9**	79.0	6.3
		OWL (Ours)	**53.4**	**64.2**	**79.9**	**55.1**	**52.5**	**53.4**	**64.4**	**64.7**	21.2	26.8	**79.0**	**6.3**

Bold values indicate the best result

The table reports *within*-dataset performance (SemanticKITTI), and *cross*-dataset performance (KITTI360) for two class ontologies, where *Vocabulary 1* merges rarer classes into a catch-all `other` class while *Vocabulary 2* closely follows original SemanticKITTI class definitions. Interestingly, OWL outperforms baselines even for `known` classes (69.4% PQ compared with 67.8% 4D-PLS in-domain and 59.4% vs. 56.1% PQ cross-domain for *Vocab. 1*). For new `unknown` classes at test-time, OWL recalls 45.1% of labeled instances (36.3% UQ) while a modified version of 4D-PLS can only recall 6.0%, leading to 4.2% UQ. We observe similar trends with *Vocab. 2*. Note that *Vocab. 1* generalizes much better across datasets, suggesting that grouping rare classes in a catch-all `other` during training leads to better generalization. These results also suggest that we do not need a separate geometric clustering module for `other` classes, as suggested in Wong et al. (2020). Such an approach is similar to OWL ‡ (which uses a learned instance grouping for `thing` classes and a separate hierarchical segmentation tree for `other`), which does not improve performance

Table 4 Per-class results (Vocabulary 2) w.r.t. PQ for known classes and UQ for the `unknown`. We observe performance drop for rarer classes (*e.g.*, `motorcycle`) and among `stuff` (*e.g.*, `building`, `vegetation`) that may look differently in different areas of the city

		Car	Hum.	Truck	Moto.	Bic.	Traf. Sign	Trunk	Pole	Rd.	Build.	Veg.	Fen.	Sidew.	Ter.	Park.	unknown
SemK.	4D-PLS (Aygün et al., 2021)	82.2	68.0	44.1	59.4	35.9	54.4	60.7	60.7	94.0	86.6	86.7	22.0	77.7	58.4	21.9	16.4
	PolarSeg-Panoptic (Zhou et al., 2021)	85.4	52.6	52.5	52.6	39.5	47.5	33.4	49.0	90.7	84.8	85.4	16.4	70.6	53.3	20.7	14.9
	OWL	90.6	68.0	50.2	65.4	40.3	54.4	51.7	60.7	94.0	86.6	86.7	22.0	77.7	58.4	21.9	49.3
K.360	4D-PLS (Aygün et al., 2021)	81.5	52.4	41.9	43.4	20.3	47.0	NA	20.9	89.8	60.6	77.2	9.6	63.0	23.5	11.3	3.4
	PolarSeg-Panoptic (Zhou et al., 2021)	1.7	0.0	0.4	0.0	0.0	2.7	NA	1.0	0.6	0.8	1.0	0.0	0.2	0.1	0.0	0.0
	OWL	88.3	62.1	55.8	45.8	23.2	74.3	NA	63.2	89.8	60.6	77.2	9.6	63.0	23.5	11.3	21.2

SemanticKITTI→ KITTI360 gap Why does performance drop from SemanticKITTI to KITTI360, even though both were recorded in the same city, with the same sensor? To answer this question, we analyze per-class performance in Table 4 (*Vocabulary 2*) and confusion tables (Fig. 5). For common `thing` classes, there is a minimal performance drop (90.6 → 88.3 PQ for `car`). Indeed *cars should look identical in different regions of the city*. As expected, we observe a performance drop for rarer `thing` classes (65.4 → 45.8 for `motorcycle`, 40.3 → 23.2 for `bicycle`), as only a handful of instances of these classes are observed in the (dominantly static) SemanticKITTI. We observe larger performance drops for `stuff` classes (*e.g.*, `building`, `vegetation`, `fence`, `terrain`). The reason for this drop is two-fold. Firstly, KITTI360 covers a larger area of the city that does not overlap with SemanticKITTI. Therefore, in KITTI360, we observe a larger diversity of these classes, which confuses the semantic classifier. Second, these classes are often confused with `stuff` classes, labeled only in KITTI360. For example, `building` is commonly confused with KITTI360 classes `garage` and `wall`, and `fence` is confused with `wall` and `gate`. Class `thrash bin` is often confused with a `stuff` class `sidewalk`, likely due to context: thrash bins are usually seen on sidewalks. Outlier synthesis (Cen et al., 2022; Kong & Ramanan, 2021) could be used to minimize this confusion in the future.

5.4 Confusion Analysis

To further analyze the per-class semantic segmentation performance, we plot extended confusion matrices, similar to those reported in open-set object detection (Dhamija et al., 2020). The horizontal axis represents the ground-truth classes, and the vertical axis represents OWL predictions. We extend the `other` class into its fine-grained split on the horizontal axis. Since *Vocabulary 1* consists of more held-out classes, we visualize the matrices for this setting.

5.4.1 SemanticKITTI

In Fig. 4, we show the confusion of OWL on SemanticKITTI. Among `known` classes, we observe that the `terrain` class is most confused with the `vegetation` class. In the `other` class, we observe significant confusion with the `known` classes. For example, `other-vehicle` is often misclassified as `car` or `truck`. Furthermore, `other-ground` and `parking` are often misclassified as `sidewalk` and `road`. Confusion most commonly arises between classes with super-classes.

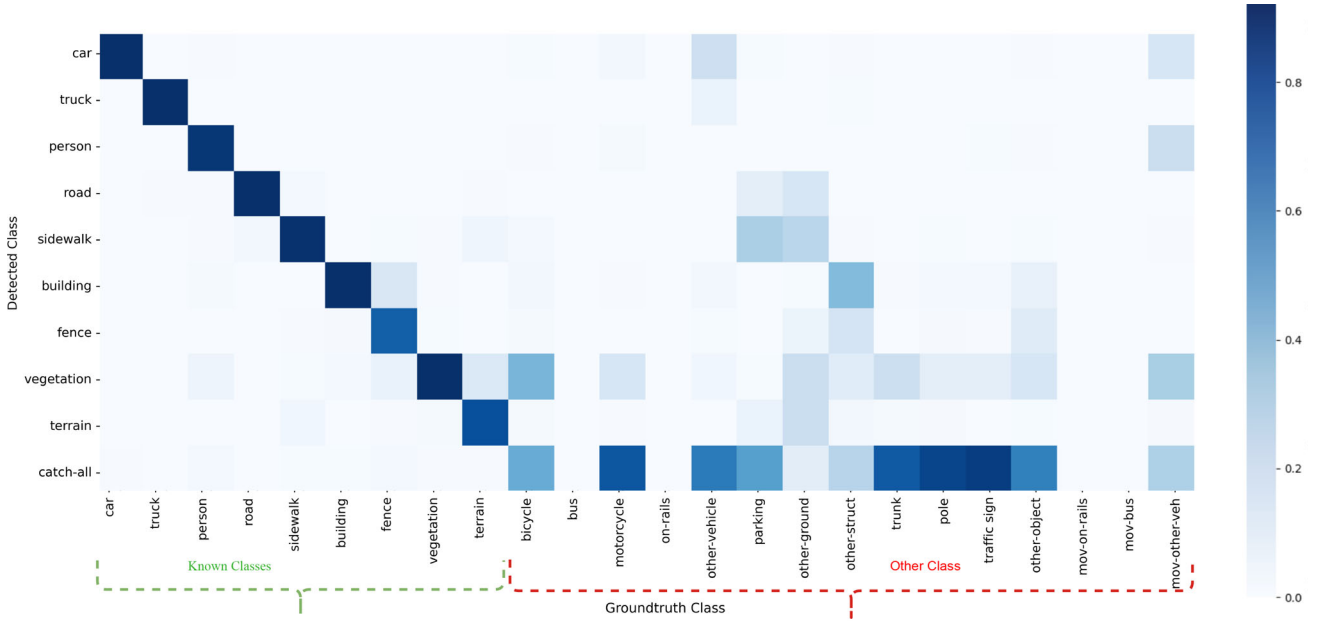


Fig. 4 The extended confusion matrix for OWL trained on SemanticKITTI and evaluated in-domain (on SemanticKITTI), using *Vocabulary 1*. On the left side, we see the confusion among `known` classes. On the right, we can see which `known` classes are confused with classes that form the `other` class. For `known` classes, we observe a confusion between (related) terrain and vegetation. We also observe that several `other` points are misclassified as `known`. Class `other-vehicle` is often misclassified as `car` or `truck`, while `other-ground` and `parking` are commonly misclassified as `sidewalk` and `road` classes. This explains the low IoU, observed in Table 2 (main paper) on `other` in SemanticKITTI (Color figure online)

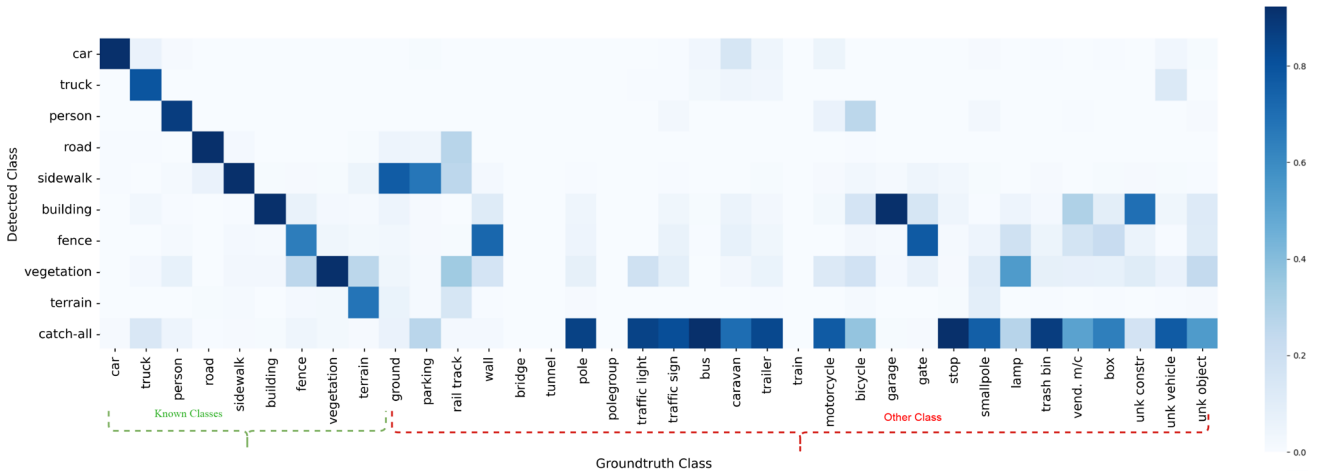


Fig. 5 The extended confusion matrix for OWL trained on SemanticKITTI and evaluated in cross-domain setting (on KITTI-360), using *Vocabulary 1*. On the left side, we see the confusion among `known` classes. On the right, we can see which `known` classes are confused with classes that form the `other` class. Contrary to the in-domain confusion, we observe more confusion within `known` classes. For instance, `car` and `truck` classes are often confused. The class `sidewalk` is often misclassified as `terrain`, while almost all `known` classes are confused with `vegetation`. As can be seen, there is confusion between `known` and `unknown` classes. `Ground` and `parking` are often predicted as `road` and `sidewalk`. Class `wall` (a novel `other-stuff` class) is confused with `fence`, building, and vegetation, presumably due to their geometric similarity. Class `trailer` is frequently confused with class `car`. As demonstrated, cross-domain semantic segmentation is a challenging problem (Color figure online)

5.4.2 KITTI360

In Fig. 5, we visualize the extended confusion matrix for OWL on KITTI360. In the `known` classes, we observe confusion between the `car` and `truck` classes. Furthermore, `fence` and `terrain` are also frequently misclassified as `vegetation`. Next, we analyze confusion between `known` and `other` classes. We observe that `ground` and `parking`

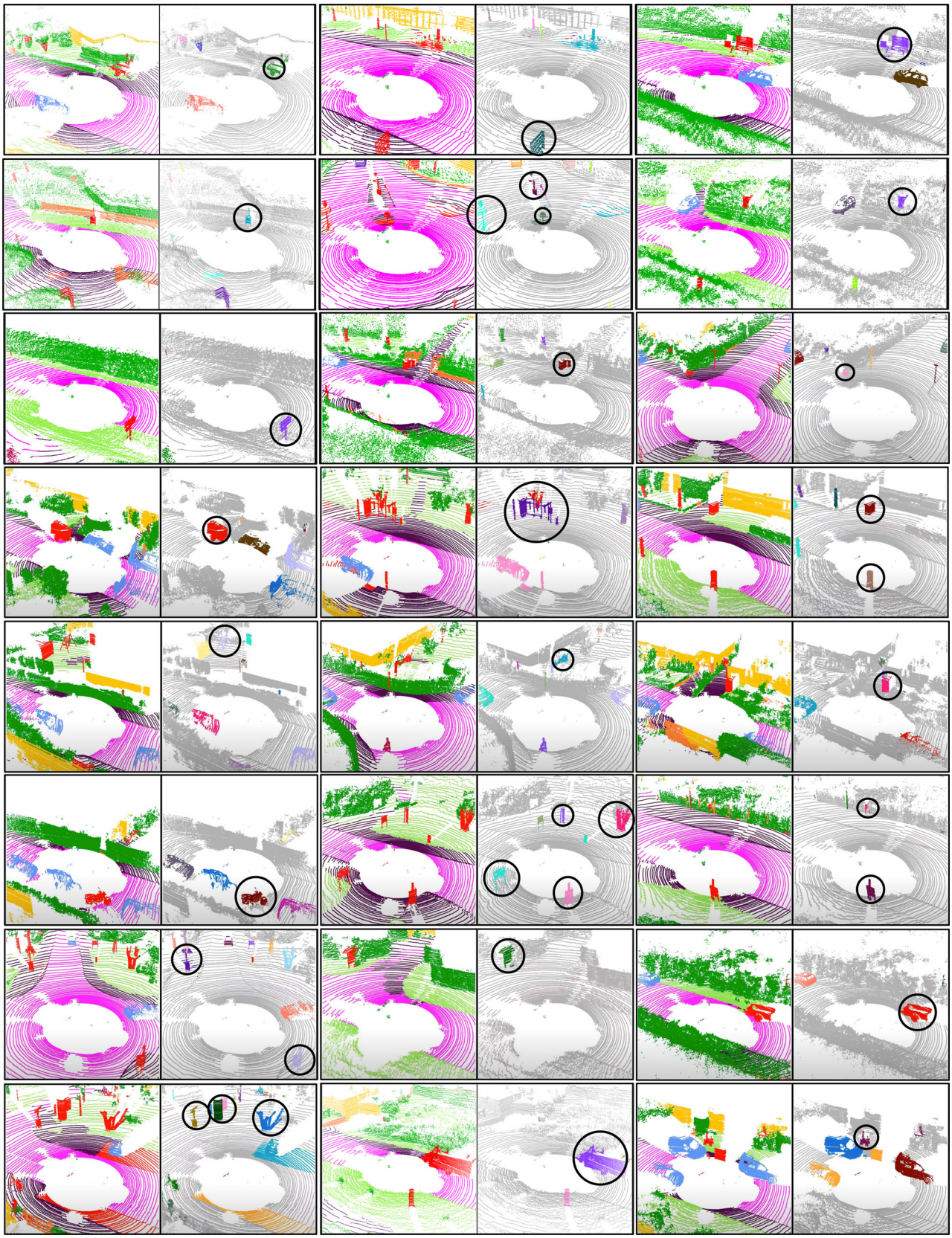


Fig. 6 Qualitative results on KITTI360 dataset. OWL successfully segments several `other` objects (*left*, shown in **red**; *right*: segmented instances). Issues and challenges: we observe OWL occasionally under-segments `other` instances (see, *e.g.*, *top* row) (Color figure online)

are often misclassified as road and sidewalk. Walls are confused with the fence, building, and vegetation, while trailer is commonly confused with car. This demonstrates the challenge of open-world generalization of semantic segmentation, indicating the need for future research on this front.

5.5 Qualitative Results

In Fig. 6, we show several visual examples for OWL in the KITTI360 dataset (*Vocabulary 1*), focusing on instances of unknown classes. As can be seen, OWL performs well in challenging cases with several unknown objects in the same scene. For example, in *sixth-row-middle* and *eighth-row-left*, our method segments common objects, such as trunk, sign board, and pole. Moreover, OWL segments several rarer objects. For example, in *first-row-left*, OWL segments a wheelbarrow. In *first-row-right*, OWL segments a bus stop. In *fifth-row-left*, *fifth-row-middle* and *sixth-row-left*, OWL segments a swing, stoller, and a motorcycle.

However, we also observe failure cases. In the top row of Fig 6, OWL sometimes under-segment nearby objects. For example, on *first-row-mid* and,*first-row-right* neighboring poles and signs are clustered as a single instance.

To further showcase the performance of our method, we provide a video with OWL inference on SemanticKITTI and KITTI360 lidar sweeps.

6 Discussion and Conclusion

We investigate Lidar Panoptic Segmentation in an Open World setting (LiPSOW), for which we set up baselines and an evaluation protocol. We demonstrate that our OWL performs significantly better than prior work for in-domain and cross-domain evaluations. In addition to better generalization across domains, OWL segments a large number of instances in the other class. Finally, we observed that grouping rare classes into a catch-all other class leads to significantly better cross-domain generalization. We hope our insights spark future investigation and help build perception models that can generalize to novel environments.

We envision LiPSOW as a first stage towards an end-to-end continual learning paradigm where unknown objects from lidar scans are discovered online and clustered offline. Based on discovered object clusters (with human-in-the-loop annotations to provide further categorical refinement), the network can be updated using incremental/continual learning. Such signals can be incorporated into trajectory prediction or motion planning algorithms to enable safe maneuvers.

A LiDAR Panoptic Segmentation in Open-World

A.1 Vocabulary Splits

We detail our vocabulary split for SemanticKITTI (Behley et al., 2019, 2021) and KITTI360 (Liao et al., 2021) in Table 5. The categorization of stuff and thing follows from the KITTI360 ontology. *Vocabulary 1* is constructed by sorting SemanticKITTI superclasses by the number of instances in a superclass and holding-out tail classes as other. *Vocabulary 2* is constructed by holding out only *the rarest* object instances that do not correspond to any semantic class, labeled in SemanticKITTI (*e.g.*, other-vehicle, other-object, *etc.*)

A.2 Vocabulary Consistency

Inconsistent labeling policies across datasets cause label shifts. For this reason, we base our evaluation set-up on SemanticKITTI and KITTI360. The two datasets adopt largely consistent labeling policies and the same sensor; therefore, the shift in data distribution can be thought of as a result of new classes emerging across datasets. To ensure we have a consistent class vocabulary, we perform the following measures:

- We merge rider and bicyclist (SemanticKITTI) with human and rider (KITTI360) into a single human class to ensure consistency.
- Classes pole and traffic sign are commonly treated as thing classes. However, in SemanticKITTI, they are treated as stuff classes because we do not have instance-level annotations for them. As instance labels for these classes are available in KITTI360, we treat them as other thing classes in KITTI360. Therefore, individual instances of these classes must be segmented. This is consistent with the overall goal of LiPSOW: methods must segment all instances, including those not labeled in SemanticKITTI.
- We treat building as a stuff class in SemanticKITTI and KITTI360 (*i.e.*, we do not treat the building class as an object, a thing class).

A.3 Model Training

We train all LiPSOW methods using instance labels for known-things and semantic labels for known- things and known-stuff. The instance labels for the other classes are held out, *i.e.*, not available during training. The LiPSOW methods should classify these points as other instead of performing a fine-grained semantic classification of these points. Additionally, LiPSOW methods need to seg-

Table 5 LiPSOW class ontology

	`Known`	`Other` (SemanticKITTI)	`Unknown` (KITTI360)
Vocab. 1	*Car, truck, human* **road, sidewalk, fence, vegetation, terrain, building**	*Bicycle, motorcycle, other-vehicle*, ***trunk, pole, traffic-sign, other-structure, other-object*** **other-ground, parking**	*Pole, pole group, traffic light, traffic sign, bus, caravan, trailer, train, motorcycle, bicycle, garage, stop, small pole, lamp, trash bin, vending machine, box, unk. construction, unk. vehicle, unk. object* **ground, parking, rail-track, wall, bridge, tunnel, gate**
Vocab. 2	*Car, bicycle, motorcycle, truck, human*, ***trunk, pole, traffic-sign*** **road, sidewalk, fence, vegetation, terrain, parking, building**	*Other-vehicle*, ***other-structure, other-object*** **other-ground**	*Traffic light, bus, caravan, trailer, train, garage, stop, lamp, trash bin, vending machine, box, unk. construction, unk. vehicle, unk. object* **ground, rail-track, wall, bridge, tunnel, gate**

In our experimental section, we report results using two vocabularies: *Vocabulary 1* & *Vocabulary 2*, both derived from the SemanticKITTI (Behley et al., 2019, 2021) class ontology. We color `stuff` classes in bold and `thing` classes in italic. With bolditalic we denote classes with instance labels available in KITTI360, but not in SemanticKITTI. LiPSOW methods have access to all labels from `known` classes; however, no access to labels for `other`

ment novel instances from `other`, *e.g.*, segment bicycles in SemanticKITTI (*Vocabulary 1*) and vending machines in KITTI360 (*Vocabulary 1* & *Vocabulary 2*).

A.4 KITTI360 Ground-Truth

To evaluate methods on the KITTI360 dataset, we require per-scan semantic and instance labels for Velodyne Lidar scans. However, KITTI360 only provides multiple accumulated point clouds (accumulated over approximately 200 m), recorded by the SICK Lidar sensor. We use these dense accumulated point clouds to retrieve per-scan labels for individual Velodyne point clouds. Concretely, we use publicly available scripts (Mosig, 2022) to align individual Velodyne scans with the accumulated point cloud based on known vehicle odometry. Once aligned, we perform a nearest neighbor search for each Velodyne point in the corresponding accumulated point cloud. In case a match is not found within a 10 cm radius, we mark this point as unlabeled (ignored during the evaluation). We visualize the retrieved labels for Velodyne point clouds in Fig. 7.

B Implementation Details

B.1 4DPLS†

Our method and several baselines are based on 4D-PLS (Aygün et al., 2021) that employs an encoder-decoder point-based KPConv (Thomas et al., 2019) backbone for point classification and instance segmentation. The instance segmentation branch consists of three network heads. The *objectness* head predicts for all points how likely they are to represent a (modal) instance center. The *embedding* and *variance* heads are used to associate points with their respective instance centers. During the inference, we select the point p_i with the highest objectness, evaluate all points under a Gaussian (parameterized by the predicted mean and variances for p_i), and assign points to this cluster if the point-to-center association probability is higher than a threshold, *i.e.*, > 0.5. This process is repeated until the maximum objectness is below a certain threshold (0.1 in 4D-PLS). To ensure high-quality segments, 4D-PLS also enforces that the highest objectness should be > 0.7.

Since the objectness head is trained independently of the semantic head in a class-agnostic fashion, we hypothesize it should be able to learn a general notion of objectness from geometric cues. Therefore, we evaluate whether it can segment instances of novel classes with lower confidence. Therefore, we adapt the inference procedure in 4D-PLS to allow additional instances to be segmented. We achieve this by reducing the minimum objectness threshold from 0.7 to 0.3 for `other`, while maintaining the same threshold for `known things`. Our experimental evaluation confirms that this baseline can segment a larger number of `other` instances compared to 4D-PLS.

B.2 OSeg (Cen et al., 2022)

OSeg (Cen et al., 2022) introduces a novel strategy for open-world semantic segmentation of LiDAR Point Clouds. The proposed framework consists of two stages: (i) Open-Set semantic segmentation (OSeg) and (ii) Incremental Learning. For a fair comparison, we benchmark OSeg against our baselines.

OSeg introduces redundancy classifiers on top of a closed-set model to output scores for the `unknown class`. In addition, OSeg uses unknown object synthesis to generate pseudo-unknown objects based on real novel objects. The OSeg formulation considers `other vehicle` as a novel category for SemanticKITTI. To benchmark under our proposed LiPSOW formulation, we modify OSeg to allow for

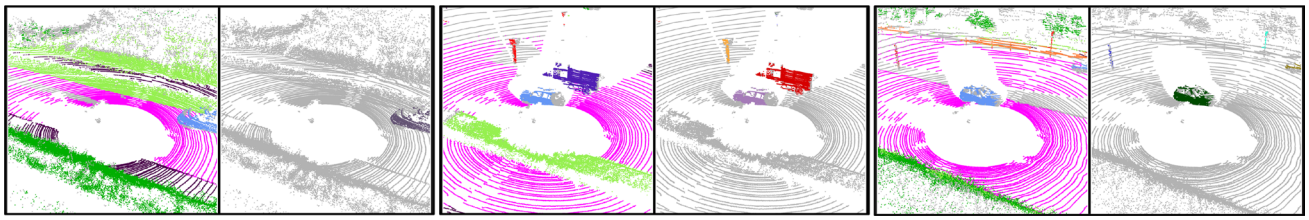


Fig. 7 Semantic and instance labels for KITTI360 evaluation, retrieved from the dense accumulated labeled point clouds. Grey color denotes points for which we could not retrieve labeled points within a 10 cm radius to transfer labels (Color figure online)

more classes in `other` (based on our vocabulary splits) and train from scratch. We use the default set of hyperparameters, and use 3 redundancy classifiers.

C OWL: Instance Segmentation

C.1 Segmentation Tree Generation

Given an input point cloud, we first make a network pass and classify points into `thing`, `stuff`, and `other` classes. Then, we construct a hierarchical tree segmentation tree T by applying HDBSCAN (McInnes et al., 2017) on points. Concretely, at each level of the hierarchy, we reduce the distance threshold ϵ (HDBSCAN connectivity hyperparameter) to obtain finer point segments. Therefore, the nodes in T contain strictly smaller and finer-grained instances in child nodes. We follow Hu et al. (2020) and use distance thresholds $\epsilon \in [1.2488, 0.8136, 0.6952, 0.594, 0.4353, 0.3221]$.

C.2 Learning Objectness-Scoring Function

Given a hierarchical tree T over the input point cloud, we need to find a node partitioning such that each point is assigned to a unique instance. Naturally, some nodes in the tree would contain high-quality segments, while others would consist of a soup of segments or overly segmented instances. To associate a metric quality for each node, we follow Hu et al. (2020) and learn a function to score each segment in the node.

C.2.1 Network Architecture

Each node in the tree consists of a segment (*i.e.*, a group of points), where the number of points may vary. We concatenate all the points in a segment to get a $N \times 3$ dimensional tensor, where N is the number of points in the segment. There are several ways how to learn such a function $f(p) \rightarrow [0, 1]$ that estimates how likely a subset of points represents an object. One approach is to estimate a per-point objectness score. Following Aygün et al. (2021), this can be learned by regressing a truncated distance $O \in \mathbb{R}^{N \times 1}$ to the nearest point center of a labeled instance (Aygün et al., 2021) on-top of decoder features $F \in \mathbb{R}^{N \times D}$. The objectness value can then be averaged over the segment $p \subset P$.

Alternatively, we can train a holistic classifier as a second-stage network. The network comprises three major components: a) input projection layer, b) segment embedding layer, and c) objectness head. In the input projection layer, we project the input point cloud to a higher dimension of $N \times 256$ by passing through two fully-connected layers. Then, we compute a per-segment embedding of dimension 512 using the embedding layer. This consists of set abstraction layers inspired by PointNet++ (Qi et al., 2017), followed by a reduction over the points.

C.2.2 Network Training

The objectness head predicts per-segment objectness, using three fully-connected layers with a hidden layer of size 256. To obtain training supervision, we pre-built hierarchical segmentation trees T_i for each point cloud i in the training set and minimize the training loss based on the signal we obtain from *matched* segments between the segmentation trees and set of labeled instances, GT_i. As described in Sec 4 of the main paper, for each node in the segmentation tree, the regressor predicts the objectness value which is supervised by the intersection-over-union of the segment with the maximal matching ground-truth instance.

Alternatively, we can also formulate the network as a classifier, where the post-softmax outputs from the network can be viewed as the quality of each segment (*i.e.*, how good or bad a segment is), and this is trained using a binary cross-entropy loss. We observe that the regression formulation empirically results in a better tree cut compared to the classifier formulation. We attribute this to the over-confident and peaky distributions resulting from the classifier. The regressor formulation benefits from a smoother distribution over objectness scores, resulting in a better tree cut. We evaluate the aforementioned variants in Table 1 in the main paper.

C.3 Inference

With a score assigned to each segment, we now need to find a global segmentation, *i.e.*, the optimal instance segmentation from an exponentially large space of possible segments. The global segmentation score is defined as the worst objectness among the individual segments in a tree cut. The optimal partition is the one that maximizes the global segmentation score. We outline this inference algorithm in Algorithm 1. Each node in the tree T constitutes a segment proposal for an object instance. For each proposal S, we score its objectness using the learned objectness function f. The segment S is deemed as an optimal node to perform a tree-cut if its objectness is greater than any of its child nodes. By design, this tree-cut algorithm ensures that each segment is assigned to a unique instance. For details about the algorithm and optimality guarantees, we refer the reader to Hu et al. (2020).

Algorithm 1 Node Partitioning given a Hierarchical Segmentation Tree

function TREE- CUT(S: Point Segment, T: Tree, f(·): Scoring Function)
  $F_S \leftarrow f(S)$ ▷ Predicted objectness for segment S
  $C_S \leftarrow$ Set of children nodes for S in T
  for S_i in C_S **do**
    $T_i \leftarrow$ subtree of T with S_i as root
    $S_i, F_i \leftarrow$ TREE- CUT(S_i, T_i, f) ▷ Score each segment in the child node
    if $F_i \leq F_S$ **then return** S, F_S
    end if
  end for
  if $min_i F_i > F_S$ **then**
    $S \leftarrow \bigcup_i S_i$
    $F_S \leftarrow \min_i F_i$
  end ifreturn S, F_S
end function

D Implementation Details

D.1 Training Encoder–Decoder Network

.To train the point-classification network on each vocabulary, we follow the training procedure from Aygün et al. (2021). We train the network for 1000 epochs with a batch size of 8. We use the SGD optimizer with a learning rate of $1e-3$ and a linear decay schedule.

D.2 Second-Stage Training

.In contrast to the encoder-decoder network, which takes an entire point cloud as input, the second-stage network requires positive and negative training instances (*i.e.*, examples of objects and non-objects) from the segmentation tree to evaluate the loss functions (regression or cross-entropy losses). To generate these instances, we first generate semantic predictions. Next, we use the points classified as `thing` or `other` to generate the segmentation tree using thresholds as described in Sect C.1. Each node in the segmentation tree is a training sample for the second-stage network. We use predictions from the encoder-decoder instead of ground-truth semantic labels, since during inference the second-stage network must be robust to misclassification errors within each node in the segmentation tree.

D.3 Training Objectness Regression Function

.To train the regressor, we need to generate the corresponding ground-truth for each segment in the generated training set. For a given segment, the target score is computed as the maximum intersection-over-union of the segment with all the ground-truth instances in the dataset. Finally, we train the network using a mean-squared error loss function with a learning rate of $2e-3$ and batch size of 512 for 200 epochs.

D.4 Training Objectness Classification Function

Alternatively, learning an objectness function can be posed as a classification problem, rather than a regression problem. In this case, we supervise the network via cross-entropy loss. The target labels for training this classifier are obtained by binarizing the regression targets using predefined intersection-over-union (IoU) thresholds. A regression target with an IoU greater than 0.7 is defined as a positive segment, and a regression target with an IoU less than 0.3 is treated as a negative segment. Since the ground-truth classification targets are generated from the hierarchical tree, which consists of predicted `known-things` or `other`, the generated training data is strongly biased towards positive samples. To elaborate, since the point classification network performs well, the segments in the tree most likely consist of `known-things` or `other`, barring misclassification error. Therefore, while training this network, we observe a disproportionate imbalance towards the positive class. To mitigate this, perform a weighted resampling; We resample instances (segments) of either positive or negative classes with a probability proportional to the inverse frequency of that class.

E Complexity Analysis

OWL requires a two-stage training process. The first stage method, 4DPLS (Aygün et al., 2021) is not a real-time. In addition, the second stage requires the construction of a segmentation tree. Given N points, the algorithm's (Hu et al., 2020) time complexity and space complexity are linear in N. In practice, we observe that N is quite large, often of the

order of 100,000+ points per scan. Therefore, a limitation of OWL is that it cannot be run in real-time.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s11263-024-02166-9.

Author Contributions Experiments were performed by Anirudh Chakravarthy and Meghana Ganesina, who were advised by Aljosa Osep, Deva Ramanan, Shu Kong, and Laura Leal-Taixé. Peiyun Hu provided support with instance segmentation implementation. All authors approved the manuscript submission.

Funding This work was supported by the CMU Argo AI Center for Autonomous Vehicle Research.

Data Availability All experiments make use of publicly available datasets (SemanticKITTI, Kitti360). 4DPLS (Aygün et al., 2021), PolarSeg-Panoptic (Zhou et al., 2021), and OSeg (Cen et al., 2022) experiments use open-source code.

Declarations

Conflict of interest The authors have no Conflict of interest to declare that are relevant to the content of this article.

Ethical Approval Not applicable.

Consent to participate Not applicable.

Consent for publication Not applicable.

Code Availability Code has been released along with the submission.

References

Alexe, B., Deselaers, T., & Ferrari, V. (2012). Measuring the objectness of image windows. *IEEE TPAMI, 34*(11), 2189–2202.

Alonso, I., Riazuelo, L., Montesano, L., & Murillo, A. C. (2020). 3d-mininet: Learning a 2d representation from point clouds for fast and efficient 3d lidar semantic segmentation. *IEEE Robotics and Automation Letters, 5*(4), 5432–5439.

Aygün, M., Osep, A., Weber, M., Maximov, M., Stachniss, C., Behley, J., & Leal-Taixé, L. (2021). 4d panoptic lidar segmentation. In *CVPR*.

Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., & Gall, J. (2019). SemanticKITTI: A dataset for semantic scene understanding of LiDAR sequences. In *ICCV*.

Behley, J., Milioto, A., & Stachniss, C. (2021). A benchmark for LiDAR-based panoptic segmentation based on KITTI. In *International Conference on Robotics and Automation*.

Behley, J., Steinhage, V., & Cremers, A.B. (2013). Laser-based segment classification using a mixture of bag-of-words. In *International Conference on Intelligent Robots and Systems*.

Bendale, A., & Boult, T.E. (2016) Towards open set deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1563–1572.

Boykov, Y., & Funka-Lea, G. (2006). Graph cuts and efficient nd image segmentation. In *IJCV***70**(2).

Cen, J., Yun, P., Zhang, S., Cai, J., Luan, D., Tang, M., Liu, M., & Yu Wang, M. (2022). Open-world semantic segmentation for lidar point clouds. In *ECCV*.

Chen, X., Kundu, K., Zhu, Y., Berneshawi, A.G., Ma, H., Fidler, S., & Urtasun, R. (2015). 3D object proposals for accurate object class detection. In *Advances in Neural Information Processing Systems*.

Choy, C., Gwak, J., & Savarese, S. (2019). 4D spatio-temporal convnets: Minkowski convolutional neural networks. In *CVPR*.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., & Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. In *CVPR*.

Dewan, A., Caselitz, T., Tipaldi, G.D., & Burgard, W. (2015). Motion-based detection and tracking in 3d lidar scans. In *International Conference on Robotics and Automation*.

Dhamija, A.R., Günther, M., & Boult, T.E. (2018). Reducing network agnostophobia. In *NeurIPS*.

Dhamija, A., Gunther, M., Ventura, J., & Boult, T. (2020). The overlooked elephant of object detection: Open set. In *Wint. Conf. App. Comput. Vis*.

Douillard, B., Underwood, J., Kuntz, N., Vlaskine, V., Quadros, A., Morton, P., & Frenkel, A. (2011). On the segmentation of 3d lidar point clouds. In *2011 IEEE International Conference on Robotics and Automation*, pp. 2798–2805. IEEE.

Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Robotics: Science and Systems*.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2010). The pascal visual object classes (VOC) challenge. *IJCV, 88*(2), 303–338.

Fomenko, V., Elezi, I., Ramanan, D., Leal-Taix'e, L., & Osep, A. (2022). Learning to discover and detect objects. In *Advances in Neural Information Processing Systems*.

Fong, W.K., Mohan, R., Hurtado, J.V., Zhou, L., Caesar, H., Beijbom, O., & Valada, A. (2021). Panoptic nuscenes: A large-scale benchmark for lidar panoptic segmentation and tracking. arXiv preprint arXiv:2109.03805

Gasperini, S., Mahani, M.-A.N., Marcos-Ramiro, A., Navab, N., & Tombari, F. (2021). *Panoster: End-to-end panoptic segmentation of lidar point clouds*. Letters: IEEE Rob. Automat.

Held, D., Guillory, D., Rebsamen, B., Thrun, S., & Savarese, S. (2016). A probabilistic framework for real-time 3d segmentation using spatial, temporal, and semantic cues. In *Robotics: Science and Systems*.

Hendrycks, D., Mazeika, M., & Dietterich, T. (2019). Deep anomaly detection with outlier exposure. In *ICLR*.

Hong, F., Zhou, H., Zhu, X., Li, H., & Liu, Z. (2021). Lidar-based panoptic segmentation via dynamic shifting network. In *CVPR*.

Hong, F., Kong, L., Zhou, H., Zhu, X., Li, H., & Liu, Z. (2024). Unified 3d and 4d panoptic segmentation via dynamic shifting networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. https://doi.org/10.1109/TPAMI.2023.3349304

Hu, P., Held, D., & Ramanan, D. (2020). Learning to optimally segment point clouds. *IEEE Robotics and Automation Letters, 5*(2), 875–882.

Hwang, J., Oh, S.W., Lee, J.-Y., & Han, B. (2021). Exemplar-based open-set panoptic segmentation network. In *CVPR*.

Jiang, P., & Saripalli, S. (2021). Lidarnet: A boundary-aware domain adaptation model for point cloud semantic segmentation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2457–2464. IEEE.

Joseph, K.J., Khan, S., Khan, F.S., & Balasubramanian, V.N. (2021). Towards open world object detection. In *CVPR*.

Kirillov, A., He, K., Girshick, R., Rother, C., & Dollár, P. (2019). Panoptic segmentation. In *CVPR*.

Klasing, K., Wollherr, D., & Buss, M. (2008). A clustering method for efficient segmentation of 3d laser data. In *2008 IEEE International Conference on Robotics and Automation*, pp. 4043–4048. IEEE.

Kong, S., & Fowlkes, C.C. (2018). Recurrent pixel embedding for instance grouping. In *CVPR*.

Kong, S., & Ramanan, D. (2021). Opengan: Open-set recognition via open data generation. In *ICCV*.

Kong, L., Quader, N., & Liong, V.E. (2023). Conda: Unsupervised domain adaptation for lidar segmentation via regularized domain concatenation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9338–9345. IEEE.

Kreuzberg, L., Zulfikar, I.E., Mahadevan, S., Engelmann, F., & Leibe, B. (2022). 4d-stop: Panoptic segmentation of 4d lidar using spatio-temporal object proposal generation and aggregation. In *ECCV AVVision Workshop*.

Langer, F., Milioto, A., Haag, A., Behley, J., & Stachniss, C. (2020). Domain transfer for semantic segmentation of lidar data using deep neural networks. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8263–8270. IEEE

Li, J., He, X., Wen, Y., Gao, Y., Cheng, X., & Zhang, D. (2022). Panoptic-phnet: Towards real-time and high-precision lidar panoptic segmentation via clustering pseudo heatmap. In *CVPR*.

Li, E., Razani, R., Xu, Y., & Liu, B. (2023). Cpseg: Cluster-free panoptic segmentation of 3d lidar point clouds. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8239–8245. IEEE.

Li, X., Zhang, G., Pan, H., & Wang, Z. (2022). Cpgnet: Cascade point-grid fusion network for real-time lidar semantic segmentation. In *2022 International Conference on Robotics and Automation (ICRA)*, pp. 11117–11123. IEEE.

Li, X., Zhang, G., Wang, B., Hu, Y., & Yin, B. (2023). Center focusing network for real-time lidar panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13425–13434.

Liao, Y., Xie, J., & Geiger, A. (2021). KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. arXiv preprint arXiv:2109.13410

Lin, Z., Pathak, D., Wang, Y.-X., Ramanan, D., & Kong, S. (2022). Continual learning with evolving class ontologies. *Advances in Neural Information Processing Systems, 35*, 7671–7684.

Liu, Y., Zulfikar, I.E., Luiten, J., Dave, A., Ramanan, D., Leibe, B., Osep, A., & Leal-Taixé, L. (2022). Opening up open world tracking. In *CVPR*.

Loiseau, R., Aubry, M., & Landrieu, L. (2022). Online segmentation of lidar sequences: Dataset and algorithm. In *European Conference on Computer Vision*, pp. 301–317. Springer.

Marcuzzi, R., Nunes, L., Wiesmann, L., Behley, J., & Stachniss, C. (2023). Mask-based panoptic lidar segmentation for autonomous driving. *IEEE Robotics and Automation Letters, 8*(2), 1141–1148.

Martin, D., Fowlkes, C., Tal, D., & Malik, J. (2001). A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*.

McInnes, L., Healy, J., & Astels, S. (2017). HDBSCAN: Hierarchical density based clustering. *Journal of Open Source Software, 2*(11), 205.

Moosmann, F., & Stiller, C. (2013). Joint self-localization and tracking of generic objects in 3d range data. In *International Conference on Robotics and Automation*.

Moosmann, F., Pink, O., & Stiller, C. (2009). Segmentation of 3d lidar data in non-flat urban environments using a local convexity criterion. In *IEEE Intelligent Vehicles Symposium*.

Mosig, C. (2022). ROS package to publish the KITTI-360 dataset. https://github.com/dcmlr/kitti360_ros_player

Najibi, M., Ji, J., Zhou, Y., Qi, C.R., Yan, X., Ettinger, S., & Anguelov, D. (2022). Motion inspired unsupervised perception and prediction in autonomous driving. In *ECCV*.

Najibi, M., Ji, J., Zhou, Y., Qi, C.R., Yan, X., Ettinger, S., & Anguelov, D. (2023). Unsupervised 3d perception with 2d vision-language distillation for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. pp. 8602–8612

Neuhold, G., Ollmann, T., Rota Bulo, S., & Kontschieder, P. (2017). The mapillary vistas dataset for semantic understanding of street scenes. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4990–4999.

Nunes, L., Chen, X., Marcuzzi, R., Osep, A., Leal-Taixé, L., Stachniss, C., & Behley, J. (2022). Unsupervised class-agnostic instance segmentation of 3d lidar data for autonomous vehicles. *IEEE Robotics and Automation Letters, 7*(4), 8713–8720.

Osep, A., Mehner, W., Voigtlaender, P., & Leibe, B. (2018). Track, then decide: Category-agnostic vision-based multi-object tracking. In *International Conference on Robotics and Automation*.

Osep, A., Voigtlaender, P., Luiten, J., Breuers, S., & Leibe, B. (2018). Towards large-scale video video object mining. In *ECCV Workshop on Interactive and Adaptive Learning in an Open World*.

Osep, A., Voigtlaender, P., Luiten, J., Breuers, S., & Leibe, B. (2019). Large-scale object mining for object discovery from unlabeled video. In *International Conference on Robotics and Automation*.

Osep, A., Voigtlaender, P., Weber, M., Luiten, J., & Leibe, B. (2020). 4d generic video object proposals. In *International Conference on Robotics and Automation*.

Oza, P., & Patel, V.M. (2019). C2AE: Class conditioned auto-encoder for open-set recognition. In *CVPR*.

Qi, C.R., Su, H., Mo, K., & Guibas, L.J. (2017). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*.

Qi, C.R., Su, H., Nießner, M., Dai, A., Yan, M., & Guibas, L.J. (2016). Volumetric and multi-view cnns for object classification on 3d data. In *CVPR*.

Qi, C.R., Yi, L., Su, H., & Guibas, L.J. (2017). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*.

Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., & Clark, J. (2021). Learning transferable visual models from natural language supervision.

Razani, R., Cheng, R., Li, E., Taghavi, E., Ren, Y., & Bingbing, L. (2021). Gp-s3net: Graph-based panoptic sparse semantic segmentation network. In *CVPR*.

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In *NeurIPS*.

Ren, P., Xiao, Y., Chang, X., Huang, P.-Y., Li, Z., Gupta, B. B., Chen, X., & Wang, X. (2021). A survey of deep active learning. *ACM Computing Surveys (CSUR), 54*(9), 1–40.

Rist, C.B., Enzweiler, M., & Gavrila, D.M. (2019). Cross-sensor deep domain adaptation for lidar detection and segmentation. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1535–1542. IEEE

Scheirer, W. J., Rezende Rocha, A., Sapkota, A., & Boult, T. E. (2012). Toward open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 35*(7), 1757–1772.

Shaban, A., Lee, J., Jung, S., Meng, X., & Boots, B. (2023). Lidar-uda: Self-ensembling through time for unsupervised lidar domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 19784–19794.

Shi, S., Wang, X., & Li, H. (2019). PointRCNN: 3D object proposal generation and detection from point cloud. In *CVPR*.

Sirohi, K., Mohan, R., Büscher, D., Burgard, W., & Valada, A. (2021). Efficientlps: Efficient lidar panoptic segmentation. *IEEE Transactions on Robotic, 38*, 1894.

Tang, H., Liu, Z., Zhao, S., Lin, Y., Lin, J., Wang, H., & Han, S. (2020). Searching efficient 3d architectures with sparse point-voxel convolution. In *ECCV*

Teichman, A., Levinson, J., & Thrun, S. (2011). Towards 3D object recognition via classification of arbitrary object tracks. In *International Conference on Robotics and Automation*.

Teichman, A., & Thrun, S. (2012). Tracking-based semi-supervised learning. *The International Journal of Robotics Research, 31*(7), 804–818.

Thomas, H., Qi, C.R., Deschaud, J.-E., Marcotegui, B., Goulette, F., & Guibas, L.J. (2019). Kpconv: Flexible and deformable convolution for point clouds. In *CVPR*.

Thorpe, C., Herbert, M., Kanade, T., & Shafer, S. (1991). Toward autonomous driving: The cmu navlab: i: Perception. *IEEE Expert, 6*(4), 31–42.

Wang, D.Z., Posner, I., & Newman, P. (2012). What could move? Finding cars, pedestrians and bicyclists in 3D laser data. In *International Conference on Robotics and Automation*.

Weng, Z., Ogut, M.G., Limonchik, S., & Yeung, S. (2021). Unsupervised discovery of the long-tail in instance segmentation using hierarchical self-supervision. In *CVPR*.

Wong, K., Wang, S., Ren, M., Liang, M., & Urtasun, R. (2020). Identifying unknown instances for autonomous driving. In *Conference on Robot Learning*, pp. 384–393. PMLR.

Wu, B., Zhou, X., Zhao, S., Yue, X., & Keutzer, K. (2019). Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 4376–4382. IEEE.

Xian, G., Ji, C., Zhou, L., Chen, G., Zhang, J., Li, B., Xue, X., & Pu, J. (2022). Location-guided lidar-based panoptic segmentation for autonomous driving. *IEEE Transactions on Intelligent Vehicles, 8*(2), 1473–1483.

Xu, J., Zhang, R., Dou, J., Zhu, Y., Sun, J., & Pu, S. (2021). Rpvnet: A deep and efficient range-point-voxel fusion network for lidar point cloud segmentation. In *CVPR*.

Yan, Y., Mao, Y., & Li, B. (2018). Second: Sparsely embedded convolutional detection. *Sensors, 18*(10), 3337.

Ye, M., Xu, S., Cao, T., & Chen, Q. (2021). Drinet: A dual-representation iterative learning network for point cloud segmentation. In: Proceedings of the IEEE/CVF international conference on computer vision, pp. 7447–7456.

Yi, L., Gong, B., & Funkhouser, T. (2021). Complete & label: A domain adaptation approach to semantic segmentation of lidar point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15363–15373

Yoshihashi, R., Shao, W., Kawakami, R., You, S., Iida, M., & Naemura, T. (2019). Classification-reconstruction learning for open-set recognition. In *CVPR*.

Yoshihashi, R., Shao, W., Kawakami, R., You, S., Iida, M., & Naemura, T. (2019). Classification-reconstruction learning for open-set recognition. In *CVPR*.

Zhan, X., Wang, Q., Huang, K.-h., Xiong, H., Dou, D., & Chan, A.B. (2022). A comparative survey of deep active learning. arXiv preprint arXiv:2203.13450

Zhang, L., Yang, A.J., Xiong, Y., Casas, S., Yang, B., Ren, M., & Urtasun, R. (2023). Towards unsupervised object detection from lidar point clouds. In *CVPR*.

Zhang, Z., Zhang, Z., Yu, Q., Yi, R., Xie, Y., & Ma, L. (2023). Lidar-camera panoptic segmentation via geometry-consistent and semantic-aware alignment. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3662–3671.

Zhao, Y., Zhang, X., & Huang, X. (2021). A technical survey and evaluation of traditional point cloud clustering methods for lidar panoptic segmentation. In *ICCV Workshops*.

Zhao, Y., Zhang, X., & Huang, X. (2022). A divide-and-merge point cloud clustering algorithm for lidar panoptic segmentation. In *International Conference on Robotics and Automation*.

Zhou, Z., Zhang, Y.,& Foroosh, H. (2021). Panoptic-polarnet: Proposal-free lidar point cloud panoptic segmentation. In *CVPR*.

Zhu, X., Zhou, H., Wang, T., Hong, F., Ma, Y., Li, W., Li, H., & Lin, D. (2021). Cylindrical and asymmetrical 3d convolution networks for lidar segmentation. In *CVPR*.

Zitnick, C.L., & Dollár, P. (2014). Edge Boxes: Locating object proposals from edges. In *ECCV*.