Unsupervised Class-Agnostic Instance Segmentation of 3D LiDAR Data for Autonomous Vehicles

Lucas Nunes Xieyuanli Chen Rodrigo Marcuzzi Aljosa Osep
Laura Leal-Taixé Cyrill Stachniss Jens Behley

***Abstract*—Fine-grained scene understanding is essential for autonomous driving. The context around a vehicle can change drastically while navigating, making it hard to identify and understand the different objects that may appear. Although recent efforts on semantic and panoptic segmentation pushed the field of scene understanding forward, it is still a challenging task. Current methods depend on annotations provided before deployment and are bound by the labeled classes, ignoring long-tailed classes not annotated in the training data due to the scarcity of examples. However, those long-tailed classes, such as baby strollers or unknown animals, can be crucial when interpreting the vehicle surroundings, e.g., for safe interaction. We address the problem of class-agnostic instance segmentation in this paper that also tackles the long-tailed classes. We propose a novel approach and a benchmark for class-agnostic instance segmentation and a thorough evaluation of our method on real-world data. Our method relies on a self-supervised trained network to extract point-wise features to build a graph representation of the point cloud. Then, we use GraphCut to perform foreground and background separation, achieving instance segmentation without requiring any label. Our results show that our approach is able to achieve instance segmentation and a competitive performance compared to state-of-the-art supervised methods.**

***Index Terms*—Semantic Scene Understanding, Deep Learning Methods**

I. Introduction

SEMANTIC scene understanding has been widely studied in the context of autonomous vehicles. Recent work aims to push different tasks in scene understanding such as semantic segmentation [43], [33], panoptic segmentation [24], [42], and instance segmentation forward, providing meaningful semantic information for autonomous systems. Such methods usually rely on learning algorithms, which require many annotated data points. Especially for autonomous driving, it is hard to generate representative labeled datasets due to the wide range of different object semantics, and it is hard to predict scenarios that can happen in such dynamic and complex environments. Current benchmark datasets try to provide enough semantic labels for the research community to evaluate approaches to deal with scene understanding. However, such datasets need to divide the semantics of complex environments into a finite number of classes for annotation. This leaves important long-tailed semantic classes unlabelled since they rarely appear on the data and are often ignored during data labeling. This can bias the learning-based methods to ignore or misclassify such long-tailed classes.

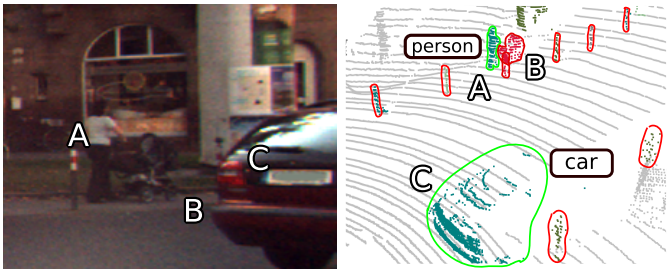

Fig. 1: We show the extended instance annotations of the SemanticKITTI dataset [3], where segments with green outline correspond to *known* classes, i.e., person *A* and car *C*, and red outlines correspond to things that are not part of the training data. While *A* and *C* are *known* classes, *B* corresponds to a baby stroller—an *unknown* class, which obviously should be detected even though not annotated in the training set.

In this paper, we address the problem of instance segmentation, more specifically, class-agnostic instance segmentation. Given the challenging task of providing enough labeled data to cover all possible semantic classes that can appear in an outdoor urban scenario, we approach the problem withdrawing the necessity of classifying the instances to compute a semantic label. Thus, the problem is reduced to identifying individual objects in the point cloud and segmenting them. The complexity of this task is caused by the unknown classes, which either do not occur in the training data or are usually ignored during the dataset annotation process but can be essential and are considered in this case.

This task can be seen as how well we can segment object instances in LiDAR point clouds when only partial knowledge about the world is given at training time, i.e., only a subset of all occurring object instances. Recent state-of-the-art approaches [20], [36] tackle this by removing the background with a predictor trained for semantic segmentation. They cluster the remaining points with different heuristics and optimization processes. Still, those learning-based methods require labeled data.

The main contribution of this paper is a new unsupervised class-agnostic instance segmentation method[1]. Given the recent developments in the field of representation learning, we use a network trained in a self-supervised manner to extract point-wise features for the point cloud and build a

Manuscript received: Feb 24, 2022; Revised: May 19, 2021; Accepted: Jun 23, 2021. This paper was recommended for publication by Editor Markus Vincze upon evaluation of the Associate Editor and Reviewers' comments.

L. N., X. C., R. M., J. B., C. S. are with the University of Bonn, Germany., A. O. and L. L. are with the Technical University of Munich, Germany.
Corresponding author: X. Chen, E-mail: xieyuanli.chen@igg.uni-bonn.de.
Digital Object Identifier (DOI): see top of this page.

[1]Code at https://github.com/PRBonn/3DUIS

graph that maps the similarities between neighboring points. Then, we use GraphCut [6] to separate the foreground objects from the background, achieving instance segmentation without requiring any labels or *semantic* background removal. Also, to evaluate the class-agnostic instance segmentation for *known* and *unknown* long-tailed classes, we propose a dataset that is an extension to the SemanticKITTI benchmark [2], [3], [17]. We provide instance labels for such long-tailed classes together with the instance labels already present in the original dataset, as exemplified in Fig. 1. Note that our dataset does not provide instance labels for long-tailed classes in the training data. However, we extend the validation and the hidden test set with additional instance labels for long-tailed classes to examine instance segmentation performance for unknown objects[2]. Our experiments show that our approach is able to segment instances without labels and without semantic background removal and achieves competitive performance compared to state-of-the-art supervised methods.

II. Related Work

In a closed-world setting, we are given full information at training time. This means that all object classes that appear at inference time are known during training. In this setting, different tasks can be defined to extract semantic knowledge in the context of autonomous driving, e.g., semantic segmentation [43], [33] or panoptic segmentation [24], [42], [23]. Large labeled datasets helped to push the state-of-the-art for learning-based approaches forward, achieving solid results. LiDAR-based methods perform well for frequently occurring objects with many training examples, e.g., cars and pedestrians. Often, however, they struggle with underrepresented classes for which few training examples exist, e.g., motorcyclists or bicyclists, and the generalization to *unknown* objects.

Self-supervised representation learning aims at learning descriptive features without labels. Given the cost of annotating data [3], such methods define simple but meaningful pretext tasks [27]. Recent works propose a discriminative task by approximating extracted features from pairs of augmented versions of one anchor image via contrastive loss function [19], [11], [18] or redundancy reduction [39]. Especially for vision transformer models [14], such methods revealed interesting properties of self-supervised learning [9], such as the ability to extract semantic information from the input data without any supervision. For point clouds, recent works using contrastive loss proposed to learn descriptive features either via scan [41] or point [37] discrimination. In our previous work [28], we proposed segment discrimination by contrasting augmented versions of segments extracted by clustering the point cloud.

In contrast to learning-based supervised methods operating on LiDAR data, bottom-up clustering methods rely on decomposing point clouds based on proximity cues via clustering. This is relevant in automotive scenarios, where intelligent vehicles need to react to all objects, including those that cannot be recognized semantically. Therefore, such methods can decompose the point cloud and define instance segments in a class-agnostic way. However, those methods generally need to consider the spatial neighborhoods of points, which can be costly to determine. To tackle that, several approaches exploit different representations such as 2D grids [34], [4], voxel grids [15], or range images [26], [5] to determine neighboring points and efficiently cluster points using these implicit neighborhoods. Besides that, several methods operate directly on the point cloud. A simple yet effective approach is to use Euclidean clustering [30]. Chen et al. [12] apply ground removal for moving instance clustering. Alternatively, Wang et al. [35] proposed to first compute a minimum spanning tree of the point cloud and learn to remove links in the graph such that the recall on the labeled classes is maximized. Our approach also uses proximity cues to define point neighborhood but mainly relies on self-supervised learned features to map the points similarities.

Recently, Hu et al. [20] proposed an algorithm that finds a segmentation from a hierarchy of multiple segmentations given a learned objectness regression function that provides an open-world scan interpretation. Such an algorithm can be used in conjunction with the aforementioned clustering methods to achieve class-agnostic instance segmentation. Wong et al. [36] proposed an open-world LiDAR panoptic segmentation method, which learns to assign points to *thing* and *stuff* classes and clusters the remaining *unknown* points using HDBSCAN [8] to obtain segmentation hypotheses for *unknown* objects. This method is evaluated on the TOR4D [38] dataset, and its subset contains only rare objects, Rare4D. Unfortunately, neither the dataset nor the proposed model is publicly available. Therefore, we believe that providing a public test-bed for unknown LiDAR instance segmentation is essential for future progress in this field. Hence, we will release our benchmark and baselines.

This paper proposes a new method based on self-supervised learned features and a graph optimization process to achieve class-agnostic instance segmentation. Unlike previous works, our approach does not rely on labeled data for training or *semantic* background removal. Therefore, it is more suitable for class-agnostic instances segmentation since it is not biased by the annotated classes, relying only on embeddings learned in an unsupervised manner.

III. Our Approach to Unsupervised Instance Segmentation

An overview of our approach is shown in Fig. 2. Our method exploits GraphCut [6] and self-supervised learned features to segment instances from the point cloud. We represent the point cloud as a graph, and use the self-supervised network to compute the points saliency map [31], [40] to sample foreground and background seeds and perform the instance segmentation. As a LiDAR point cloud may contain many instances, it can be hard to define foreground and background. Therefore, we divide ground and non-ground points and cluster the non-ground points to define the initial instances proposal. Then, we iterate over each proposal and select a cubic region of interest around it with the same size as the proposal plus a pre-defined margin. For a region of interest around the instance proposal, we define the points from the current instance as

[2]See the competition at https://bit.ly/39VFTRD

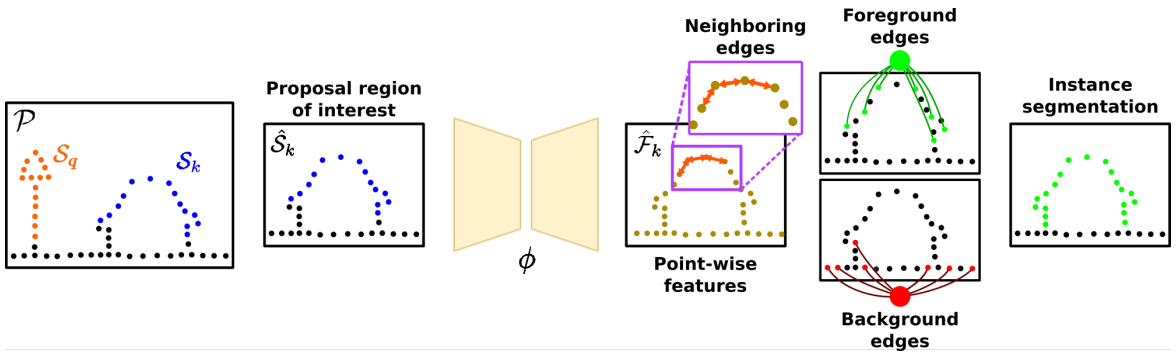

Fig. 2: Given a point cloud $\mathcal{P}$ and set of instance proposals $\mathcal{I}$, we iterate over each proposal $\mathcal{S}_k$ defining a region of interest $\hat{\mathcal{S}}_k$ around the proposal. We extract point-wise features $\hat{\mathcal{F}}_k$ for this proposal region using a network ϕ pre-trained with SegContrast [28]. Then, we build a graph weighting the neighborhood edges with the features affinity, and the foreground and background edges given the sampled seeds. Finally, we apply a min-cut over the graph to segment the instance from the background.

foreground and all the other points (from the ground or other instances) as background. Then, we create a graph from this region of interest and divide the graph into foreground and background.

A. Instance Proposals

To segment the instances, our method relies on an initial guess, so called proposals, which are refined to a more accurate segmentation. To define the instance proposals, we follow the same process used in our prior work, called SegContrast [28]. Given a point cloud $\mathcal{P} = \{\boldsymbol{p}_1, \dots, \boldsymbol{p}_N\}$ with $|\mathcal{P}| = N$ points $\boldsymbol{p}_i \in \mathbb{R}^3$. We use a ground segmentation method [22] to partition the scene into ground $\mathcal{G}$ and non-ground points $\mathcal{P}'$. Then, we cluster the non-ground points $\mathcal{P}'$ with HDBSCAN clustering [8] to divide the scan into m segments which will be the instance proposals $\mathcal{I} = \{\mathcal{S}_1, \dots, \mathcal{S}_m\}$ where $\mathcal{S}_k \subset \mathcal{P}'$ and $\mathcal{S}_k \cap \mathcal{S}_l = \emptyset, l \neq k$.

As displayed in Fig. 3, these segments are a starting point but clearly not perfect, often showing under- or over-segmentation. These segments are just the initial guess for our method. Our approach refines these proposals by mapping them in a graph of feature similarities for points inside the proposal. Then, we separate the object from the background to achieve a refined instance segmentation.

B. Self-Supervised Features

Our approach takes advantage of self-supervised representation learning to extract point-wise features. We use the MinkUNet model [13] as a feature extractor, which uses sparse convolutions, pre-trained with SegContrast [28]. SegContrast [28] relies on segments extracted in an unsupervised manner by removing the ground and clustering the remaining points. Then, the contrastive loss function is applied segment-wise, learning a feature space via segments discrimination.

For each instance proposal $\mathcal{S}_k \in \mathcal{I}$, we define a region of interest $\hat{\mathcal{S}}_k \subset \mathcal{P}$ and use the SegContrast pre-trained model ϕ to extract point-wise features $\hat{\mathcal{F}}_k$. We use those features to later compute the point similarity instead of using the directly measured information, i.e., coordinates or intensity. Since SegContrast trains the network in an unsupervised manner

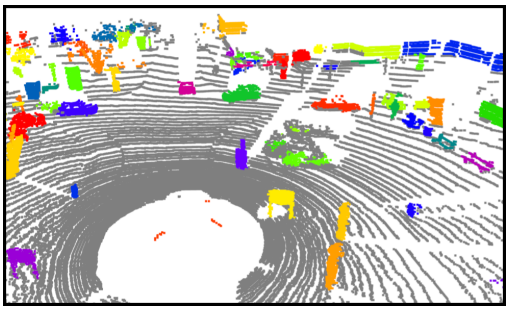

Fig. 3: Instance proposals from ground removal and clustering with HDBSCAN [8]. Different instance proposals are shown with randomly assigned colors. Best viewed in color.

to produce embeddings able to discriminate segments, such a model is likely to produce relevant features for our instance segmentation task. Therefore, we have descriptive features to identify the differences between the instance segment and the background.

C. Saliency Maps

A saliency map is commonly used to analyze the feature space learned by a network [31], [40]. To compute the saliency map, we calculate the gradients over one value to visualize the regions that have the most influence on its computation. Recently, similar visualizations were drawn from the attention layers used in Transformers [9], showing a good indication of objects boundaries even without labels [9]. Even though SegContrast [28] relies on sparse convolutional neural networks, we observed that it can also arrive at a similar "instance attention" in the saliency maps. The saliency values can be interpreted as which points are more likely to be foreground or background in the graph, which we exploit to sample the respective seeds for the GraphCut.

Given a model pre-trained with SegContrast ϕ, a point cloud $\mathcal{P}$, and an instance proposal $\mathcal{S}_k \subset \mathcal{P}$, we partition a region of interest $\hat{\mathcal{S}}_k \subset \mathcal{P}$ from the point cloud around the proposal $\mathcal{S}_k$ where $\mathcal{S}_k \subset \hat{\mathcal{S}}_k$. Then, we extract point-wise features $\hat{\mathcal{F}}_k = \{\phi(s_k) \mid s_k \in \hat{\mathcal{S}}_k\}$, and calculate the mean value δ of all the feature vectors $\hat{\mathcal{F}}_k = \{\boldsymbol{f}_{k1}, \dots, \boldsymbol{f}_{km}\}$ from the

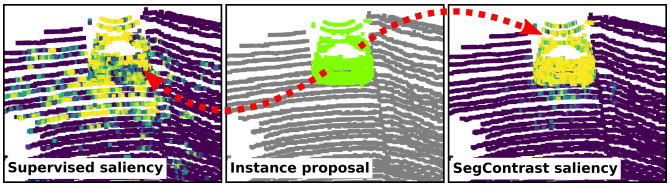

Fig. 4: Saliency map over the proposal points features, comparing the semantic segmentation supervised network and the network self-supervised trained with SegContrast. Best viewed in color.

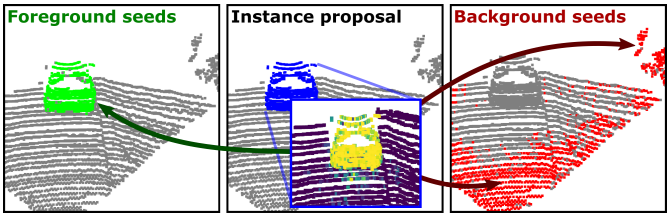

Fig. 5: From the proposal points (in blue) we compute the saliency and sample foreground (in green) and background seeds (in red).

points inside the instance proposal $\mathcal{S}_k$:

$$\delta = \frac{\sum_j^m \overline{\boldsymbol{f}}_{kj}}{m}, \quad \text{where} \quad \overline{\boldsymbol{f}}_k = \frac{\sum_i^n \boldsymbol{f}_{ki}}{n} \tag{1}$$

We compute the gradients w.r.t. δ calculated in the equation above to get the saliency map around the input proposal region $\hat{\mathcal{S}}_k$. In Fig. 4, we compare the saliency maps generated from the network trained in an unsupervised manner with SegContrast, and trained with labels for semantic segmentation, as used for background removal by Hu et al. [20]. From this comparison, we notice that the saliency from SegContrast highlights the instance points, while the network trained supervised for semantic segmentation highlights many regions around the scan, e.g., road and sidewalk. Such analysis suggests that together with the network point-wise features, we can use the saliency values to select the seeds to perform GraphCut, since it indicates the points most related to the instance.

D. Seeds Sampling

To select the foreground and background seeds, we use the instance proposals $\mathcal{S}_k$ and their saliency values. The instance proposals correspond to different objects, which means that their shape and size are variable. Sampling a fixed number of seed points may not work well for all the proposals. Thus, we sample a number of seeds according to the proposal size to perform the GraphCut. Given the proposal $\mathcal{S}_k$ and the cube region of interest around it $\hat{\mathcal{S}}_k$, we count the number of proposal points n_f and the number of non-proposal points n_b. Then, we select $\tau_f = \frac{n_f}{\gamma_f}$ points as foreground seeds and $\tau_b = \frac{n_b}{\gamma_b}$ points as background seeds, where γ_f and γ_b are pre-defined parameters.

Fig. 5 displays the seed selection process. As described in Sec. III-C, we compute the saliency map around the proposal region of interest to evidence the points most related to the instance. We select the τ_f points with the highest saliency as foreground, and the τ_b points with the lowest saliency as background. We also sample the k nearest neighbors for each foreground seed to reinforce the foreground likelihood in the seeds neighborhood. Lastly, to avoid possible outliers from the saliency map, we remove any foreground seeds from outside the proposal $\mathcal{S}_k$ and any background seeds from inside it. By doing so, we can avoid a wrong seed being improperly assigned and harming the GraphCut performance.

E. GraphCut

To segment the instances from the point cloud, we use GraphCut [6], a classical method used previously for image segmentation, applying it for point clouds. The method consists of a graph representation, mapping the relationship between each node and its neighbors and two terminal nodes, foreground and background. Then, a min-cut over the graph is applied, cutting the edges with the weakest relationship. We use the SegContrast point-wise features to compute the neighboring points similarity and define the non-terminal edges. And we calculate the saliency maps to sample the seeds and determine the edges between the points and the terminal nodes.

In our formulation, the graph contains a set of nodes $\mathcal{Z} = \{z_1, \ldots, z_{n+2}\}$, $|\mathcal{Z}| = n + 2$ where each node is a point in the proposal region $\hat{\mathcal{S}}_k$, corresponding to n points and 2 virtual terminal nodes, the foreground and background.

*1) **Terminal Edges**:* Every point has an edge with both virtual terminal nodes, and we weight these edges based on the probability of a node belonging to these nodes. Each edge probability is initialized with a small ϵ close to zero. Then, the edge probability between a selected seed and the corresponding terminal node is updated to 1.0. Finally, the edge between a node i and a terminal node t is weighted as:

$$w_{i,t} = -\lambda log(p_{i,t}), \tag{2}$$

where λ is a pre-defined parameter and $p_{i,t}$ is the defined probability for the node i to belong to the terminal node t.

*2) **Non-terminal Edges**:* For non-terminal nodes, edges are defined between each point and its k nearest neighbors, according to the coordinates of the points. To weight the edges, we use our point-wise features and calculate the dissimilarity between one point feature and its neighbors. Since GraphCut relies on the min-cut of the graph, the dissimilarity between the points should be as significant as possible. Thus, we use the L1 norm as a dissimilarity function between the point features $\boldsymbol{f}_i$ and $\boldsymbol{f}_j$ as:

$$d_{i,j} = \sum_l |\boldsymbol{f}_{il} - \boldsymbol{f}_{jl}|. \tag{3}$$

Then, the edge weight between two points will be their affinity computed given the dissimilarity $d_{i,j}$,

$$w_{i,j} = \omega \exp\left(-\frac{1}{2\sigma} d_{i,j}\right), \tag{4}$$

where σ and ω are pre-defined parameters. With the graph built as described and edges computed between each point and the terminal nodes, we then perform the min-cut on the graph, separating the instance from the background.

We do this process iteratively over each proposal to achieve the instance segmentation for the whole point cloud, having separated each object in the point cloud.

Object class	Car	Bicycle	Person	Other vehicle	Motor-cycle	Bicyclist	Truck	Motor-cyclist	**Unknown**
#instances	5,034	549	373	138	82	80	50	9	**3,587**
#bound. boxes	318,718	50,440	25,287	8,655	8,167	4,296	2,596	490	**292,871**

TABLE I: Instance and bounding box counts per class for Open-World SemanticKITTI test set. *known* instance class annotations are also annotated in the train set and *unknown* are only available on the test set and validation set.

IV. Open-World LiDAR Instance Segmentation Benchmark

This section, outlines our new and benchmark Open-World SemanticKITTI, which extends the test set and validation set of SemanticKITTI [2] with *unknown* class instance labels.

A. Problem Setting

Existing instance segmentation models assume all object classes present during inference to be manually labeled and present at training time. We refer to these object classes as *known* classes. In this paper, we also want to focus on instance segmentation for those objects that may only appear during the inference, which we denote as *unknown* classes.

More formally, the set of all object classes $\mathcal{X}$ is potentially large, and many instances occur rarely. In practice, we cannot record, label, and evaluate performance on all possible object classes, as these appear in the long tail of the object class distribution. It is thus practically only feasible to label a fixed subset of these classes.

To this end, we perform the following division. We use the labels provided from SemanticKITTI for the set of *known* classes (i.e., $\mathcal{K} = \{k_1, \ldots, k_N\} \subset \mathcal{X}$) for the whole dataset (train, validation, and test set). This set contains the frequently occurring object classes, such as *car*, *person*, *truck*, and similar. We label an additional, disjoint set of object instances only in the validation and test set, i.e., *unknown* object instances $\mathcal{U} \subset \mathcal{X}$ with $\mathcal{U} \cap \mathcal{K} = \emptyset$. Thus, our test set provides a proxy for evaluating performance for unknown objects that only appear rarely. We note that examples of these instances may appear in the training and validation set, but are not labeled as instances.

As we tackle instance segmentation, we label only *thing* classes. In literature [21], *stuff* classes are considered uncountable classes, e.g., vegetation or road. On the other hand, *thing* classes, such as car and pedestrian, usually have clearly defined boundaries, are visible in individual scans, and are countable.

B. Evaluation

For open-world instance segmentation, we evaluate how well we can decompose a LiDAR point cloud into a unique set of object instances. To quantify the performance, one possibility would be to adopt the recall-based variant of Panoptic Quality (PQ), Unknown Quality (UQ) [36], that replaces the recognition quality (RQ) term (F1-score) with a recall-based measure. However, this metric treats the stuff regions of the point cloud as a single instance, which is not desirable [29]. For example, the vegetation class could be decomposed into several instances (tree trunks, small bushes) – these are not labeled by human annotators but can be considered valid instances, depending on the task, e.g., for segment-based LiDAR odometry or SLAM [16].

We instead adopt the recently introduced LiDAR Segmentation and Tracking Quality (LSTQ) metric [1]. It consists of two terms, a classification term S_{cls} and a segmentation term S_{assoc}. However, since we propose a class-agnostic task, we remove the S_{cls} term, relying only on the S_{assoc} term to evaluate how good are the instance segments.

The association term S_{assoc} measures how well we assign points to their instances independent of the semantics:

$$S_{assoc} = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \frac{1}{|t|} \sum_{s \in \mathcal{S}, s \cap t \neq 0} TPA(s,t) IoU(s,t), \quad (5)$$

where the IoU term for each ground truth object $t \in \mathcal{T}$ and a prediction $s \in \mathcal{S}$ pair is computed based on sets of true positive associations (TPA), false negative associations (FNA), and false positive associations (FPA). These sets are evaluated in a class-agnostic manner, but differ from the work of Aygün et al. [1], as the sets are calculated per-scan instead of the whole sequence. Intuitively, the TPA set quantifies how many points were correctly assigned to their corresponding instance, and TPA and FPA sets signal two different types of point-to-instance association errors. More precisely, the TPA set contains all mutually overlapping points. The FPA set denotes all points in s that do not overlap with t, and finally, FNA contains all points from t that are not contained in s (see also Aygün et al. [1]).

The association term is class-agnostic and only informs us *how well we assign points to labeled object instances*. This allows us to evaluate instance segmentation independent of semantics, making this metric uniquely suitable for open-world LiDAR instance segmentation evaluation.

C. Open-World SemanticKITTI Dataset and Benchmark

A natural basis for a dataset suitable for benchmarking segmenting objects that appear in the long tail of the object class distribution would be a dataset that provides instance labels for the most common objects, such as traffic participants, and semantic labels for stuff classes. SemanticKITTI [3] or nuScenes-lidarseg [7] provides such labels, opposed to recent object detection datasets which provide only 3D bounding boxes, e.g., [32], [7], [10]. We opt for the SemanticKITTI dataset [3] that extends the KITTI odometry benchmark [17] with dense point-wise semantic and instance labels for each LiDAR scan. It contains $23{,}000$ labeled scans in train and $20{,}000$ labeled scans in test set, providing semantic and object instance labels for $6{,}315$, in total $418{,}649$ bounding boxes, object instances belonging to several *known* object classes.

We build the open-world SemanticKITTI benchmark, which is suitable for assessing the performance in the open-world setting. We extend the hidden test set of SemanticKITTI with $3{,}587$ instances, in total $292{,}871$ additional object instance

	S_{assoc}		
Method	*known*	*unknown*	all
Euclidean	0.651	0.426	0.611
Quick shift	0.212	0.406	0.246
HDBSCAN	0.660	0.601	0.650
Range image	0.270	0.427	0.297
Ours†	0.677	**0.605**	0.664
Ours	**0.720**	0.599	**0.699**
4D-PLS	**0.795**	0.004	0.657
Hu et al.	0.697	0.587	0.678

TABLE II: Evaluation results with S_{assoc} metric on open-world instance segmentation benchmark test set for *known*, *unknown* and all the instances. The gray rows indicate the supervised methods. Ours† is our method with ground removed using the proposals ground segmentation.

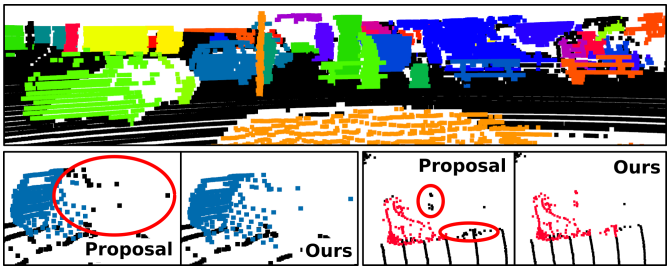

Fig. 6: Without using any background removal our method can define the boundaries between the objects and the ground. On the lower part we compare our final result with the proposal generated with HDBSCAN. Our method can refine the proposals and correctly segment points previously ignored (highlighted in the red circles).

labels for object classes that do not necessarily belong to a semantic class from the original object classes – *unknown* objects. See Tab. I for statistics on the distribution of classes. We additionally also label instances of *unknown* classes in the validation set, but provide only a way of evaluating the performance with a server-side evaluation to keep the instances unknown at training time.

V. Experimental Evaluation

The main focus of this work is a class-agnostic instance segmentation. Unlike previous methods, our work does not need instance or semantic labels to remove the background. We present our experiments to show the capabilities of our method and that it is able to segment instances without semantic background removal and achieves competitive performance compared to state-of-the-art supervised methods.

In all our experiments, we use the same parameters, defined empirically. For the instance proposal, we use a pre-defined margin of 1 meter around the proposal size to define the region of interest. For the seed sampling, we use $\gamma_f = 2$, and $\gamma_b = 2$. For GraphCut, we use $\sigma = 1.0$, $\omega = 10.0$ and $\lambda = 0.1$, and we select the $k = 8$ nearest neighbors to define the non-terminal edges, and build the graph. For the self-supervised pre-trained model, we use the same pre-training described in SegContrast [28], trained for 200 epochs.

We compare our approach in two setups: using only the segmentation from GraphCut, named as *Ours*, and with a post-processing step, where we filter points from the ground segmentation used to generate the proposals, named as *Ours*†. We compare our results with different unsupervised clustering methods and supervised learning-based methods. For the clustering methods, we use the same instance proposals process used for our approach, i.e., remove the ground, then cluster the remaining points. We evaluate HDBSCAN [8] and Euclidean clustering [30]. Also, we compare with a technique that operates on the birds-eye-view representation of the point cloud, clustering with quick shift algorithm [25], and a fast range image-based proposal generation method [5]. Additionally, we evaluate two supervised learning-based approaches. The method proposed by Hu et al. [20] removes the background with a semantic segmentation network, and cluster the remaining points given a learned *objectness* value. And a fully data-driven method 4D-PLS [1] for closed-world instance segmentation.

A. Association Quality

This experiment evaluates the association between the predicted and ground truth instances on the test set. The results show that our method achieves state-of-the-art performance, even though not using labels.

Tab. II shows the evaluation of the different methods with the S_{assoc} metric, for the *known* and *unknown* instances, and the class-agnostic case, i.e., both *known* and *unknown*. Comparing the different clustering methods, HDBSCAN presents the best performance, which supports our choice of using it for the instance proposal. The 4D-PLS method achieves the best performance for the *known* instances, while our method has the best performance for the unsupervised methods. Our approach improves by a large margin the HDBSCAN instance proposal and even surpasses Hu's method. For the *unknown* instances, the performance of supervised methods degrades. Our method achieves the best performance for *unknown* when removing the ground points. When looking at the class-agnostic case, our method with ground removal is better than unsupervised methods, and without removing ground it is also better than supervised methods.

Tab. III shows the S_{assoc} for each *known* class. The performance of the supervised methods presents the best performance on the most represented class, i.e., car. For other classes, their performance degrades since those classes have fewer samples in the training set. Our method surpasses HDBSCAN instance proposals and the supervised methods on most classes. By using unsupervised learned features, our approach is less overfitted to frequently occurring classes, being more suited to class-agnostic instance segmentation. Fig. 6 shows our results on one scan and compares our segmentation with the HDBSCAN instance proposals.

B. Instance Segmentation Quality

This experiment evaluates the intersection-over-union (IoU) and recall between predicted instances and the ground truth. We compute the IoU and recall filtered by different IoU thresholds and calculate the average of the different thresholds to evaluate the overall performance of the methods.

Tab. IV shows the results for IoU and recall for different thresholds. With a high threshold, i.e., IoU_{90}, the supervised methods perform better than the others. Since it is a panoptic

	S_{assoc}							
Method	car	bicycle	motorcyle	truck	other-vehicle	person	bicyclist	motorcyclist
HDBSCAN	0.695	0.178	0.497	0.690	0.695	0.674	0.772	0.757
Ours†	0.714	0.193	0.508	0.693	0.698	0.678	0.769	0.747
Ours	**0.762**	**0.203**	**0.552**	**0.722**	**0.737**	**0.684**	**0.788**	**0.773**
4D-PLS	**0.876**	0.183	0.436	0.553	0.495	0.504	0.687	0.649
Hu et al.	0.755	0.200	0.539	0.512	0.509	0.503	0.693	**0.793**

TABLE III: Evaluation results with S_{assoc} on open-world instance segmentation benchmark **test set** for each *known* instance class. The gray rows indicate the supervised methods.

	IoU_{90}		$Recall_{90}$		IoU_{70}		$Recall_{70}$		IoU_{50}		$Recall_{50}$			
Method	*known*	*unknown*	*known*	*unknown*	*known*	*unknown*	*known*	*unknown*	*known*	*unknown*	*known*	*unknown*	IoU	Recall
Euclidean	**0.484**	0.263	**0.503**	0.276	0.637	0.393	0.688	0.434	0.681	0.429	0.761	0.493	0.569	0.618
Quick shift	0.064	0.221	0.067	0.232	0.119	0.347	0.136	0.385	0.184	0.396	0.247	0.468	0.158	0.185
HDBSCAN	0.424	0.388	0.444	0.402	0.645	0.565	0.713	0.617	0.710	0.605	0.819	0.683	0.594	0.657
Range image	0.100	0.213	0.106	0.225	0.223	0.408	0.257	0.462	0.278	0.455	0.349	0.541	0.236	0.274
Ours†	0.459	**0.402**	0.480	**0.415**	0.663	**0.569**	0.728	**0.619**	0.721	**0.608**	0.823	**0.683**	0.612	0.672
Ours	0.461	0.381	0.481	0.393	**0.686**	0.546	**0.755**	0.595	**0.745**	0.591	**0.850**	0.668	**0.624**	**0.686**
4D-PLS	**0.667**	0.003	**0.683**	0.003	**0.753**	0.003	**0.788**	0.004	**0.792**	0.004	**0.854**	0.004	0.613	0.644
Hu et al.	0.510	**0.410**	0.528	**0.424**	0.670	0.562	0.724	0.608	0.721	0.596	0.806	0.664	**0.624**	0.676

TABLE IV: Evaluation results with IoU and Recall with different IoU thresholds and the performance averaged over all thresholds {0.5, 0.6, 0.7, 0.8, 0.9} on **test set**. The gray rows indicate the supervised methods.

	S_{assoc}		IoU	
Method	*known*	*unknown*	*known*	*unknown*
HDBSCAN	0.660	0.601	0.607	0.531
Ours†	0.677	**0.605**	0.628	**0.538**
Ours	**0.720**	0.599	**0.646**	0.517

TABLE V: Comparison between the instance proposals of our method and our results without and with ground removal post-processing step. Evaluation on open-world instance segmentation benchmark **test set**. The gray rows indicate the supervised methods.

segmentation method, 4D-PLS achieves the best performance for the *known* instances over all the thresholds. However, it fails in segmenting *unknown* instances. With a lower threshold, our method performance gets closer to the supervised methods and surpasses them in the *unknown* instances. On the overall evaluation, our method achieves an IoU on par with supervised methods and best performance in terms of recall.

C. Limitations

We additionally compare our method with and without the ground removal post-processing to discuss the limitations. As seen in the previous section, without ground removal post-processing our method achieves the best overall performance, especially for *known* instances. However, it performs better on *unknown* instances with the ground removal.

Tab. V compares both setups with the HDBSCAN proposals. Even though removing the ground points leads to better instance segmentation for *unknown* instances, it has a higher impact for *known* instances, decreasing the performance by a large margin. In Fig. 7, we illustrate one example to explain such behavior. Since the ground segmentation is done by an unsupervised method, it may have points wrongly assigned as ground and non-ground. Therefore, our method can improve the segmentation by correctly assigning instance points previously erroneously considered ground. However, some proposals may have ground considered as instance points due to the imperfect ground segmentation. In this case, our method may sample ground points as foreground seeds, leading to segmenting a wider ground region as part of the instance. By removing the ground with the post-processing, we filter the ground region sampled as foreground, but also filter the points correctly assigned as instances points that were previously wrongly labeled as ground. Therefore, the main limitation of our method relies on the initial proposals used for sampling the foreground and background seeds.

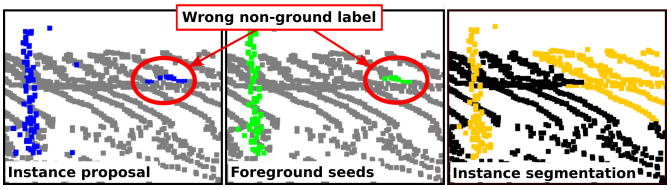

Fig. 7: Sometimes ground is wrongly assigned to an instance proposal (blue), we sample those ground points as foreground (green), affecting our final instance segmentation (orange).

VI. Conclusion

In this paper, we presented a novel approach for unsupervised class-agnostic instance segmentation. Our method uses unsupervised clustering to define instance proposals and a graph optimization algorithm to refine those proposals to more appropriate instance segmentation. Our method represents the point cloud as a graph, and exploits self-supervised representation learning to extract point-wise features to map the points neighborhood similarities in the graph. This allows us to separate the foreground instance from the background points without requiring labels. Besides, we also propose a new open-world dataset to evaluate class-agnostic instance segmentation for *known* and *unknown* instances classes and an evaluation procedure for this benchmark. The experiments suggest that our method is more suited for class-agnostic instance segmentation since it achieves competitive performance with state-of-the-art supervised methods, even surpassing it for *unknown* classes. We hope our work motivates further studies on self-supervised instance segmentation, and our benchmark contributes to the research community.

REFERENCES

[1] M. Aygun, A. Osep, M. Weber, M. Maximov, C. Stachniss, J. Behley, and L. Leal-Taixe. 4D Panoptic LiDAR Segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[2] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, J. Gall, and C. Stachniss. Towards 3D LiDAR-based semantic scene understanding of 3D point cloud sequences: The SemanticKITTI Dataset. *Intl. Journal of Robotics Research (IJRR)*, 40(8-9):959–967, 2021.

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

[4] J. Behley, V. Steinhage, and A. Cremers. Laser-based Segment Classification Using a Mixture of Bag-of-Words. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

[5] I. Bogoslavskyi and C. Stachniss. Fast range image-based segmentation of sparse 3d laser scans for online operation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2016.

[6] Y. Boykov and G. Funka-Lea. Graph cuts and efficient n-d image segmentation. *Intl. Journal of Computer Vision (IJCV)*, 70(2):109–131, 2006.

[7] H. Caesar, V. Bankiti, A. Lang, S. Vora, V. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuScenes: A Multimodal Dataset for Autonomous Driving. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[8] R.J. Campello, D. Moulavi, and J. Sander. Density-based clustering based on hierarchical density estimates. In *Proc. of the Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, 2013.

[9] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. Emerging properties in self-supervised vision transformers. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2021.

[10] M. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, and J. Hays. Argoverse: 3D Tracking and Forecasting with Rich Maps. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[11] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A Simple Framework for Contrastive Learning of Visual Representations. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2020.

[12] X. Chen, B. Mersch, L. Nunes, R. Marcuzzi, I. Vizzo, J. Behley, and C. Stachniss. Automatic Labeling to Generate Training Data for Online LiDAR-Based Moving Object Segmentation. *IEEE Robotics and Automation Letters (RA-L)*, 7(3):6107–6114, 2022.

[13] C. Choy, J. Gwak, and S. Savarese. 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[14] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2021.

[15] B. Douillard, J. Underwood, N. Kuntz, V. Vlaskine, A. Quadros, P. Morton, and A. Frenkel. On the Segmentation of 3D LIDAR Point Clouds. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2011.

[16] R. Dubé, A. Cramariuc, D. Dugas, J. Nieto, R. Siegwart, and C. Cadena. SegMap: 3d segment mapping using data-driven descriptors. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2018.

[17] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[18] J.B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko. Bootstrap Your Own Latent - A New Approach to Self-Supervised Learning. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2020.

[19] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum Contrast for Unsupervised Visual Representation Learning. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[20] P. Hu, D. Held, and D. Ramanan. Learning to Optimally Segment Point Clouds. *IEEE Robotics and Automation Letters (RA-L)*, 5(2):875–882, 2020.

[21] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár. Panoptic Segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[22] H. Lim, O. Minho, and H. Myung. Patchwork: Concentric zone-based region-wise ground segmentation with ground likelihood estimation using a 3d lidar sensor. *IEEE Robotics and Automation Letters*, 2021.

[23] R. Marcuzzi, L. Nunes, L. Wiesmann, I. Vizzo, J. Behley, and C. Stachniss. Contrastive Instance Association for 4D Panoptic Segmentation using Sequences of 3D LiDAR Scans. *IEEE Robotics and Automation Letters (RA-L)*, 7(2):1550–1557, 2022.

[24] A. Milioto, J. Behley, C. McCool, and C. Stachniss. LiDAR Panoptic Segmentation for Autonomous Driving. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[25] D. Mitzel and B. Leibe. Taking mobile multi-object tracking to the next level: People, unknown objects, and carried items. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2012.

[26] F. Moosmann, O. Pink, and C. Stiller. Segmentation of 3D Lidar Data in non-flat Urban Environments using a Local Convexity Criterion. In *Proc. of the Intelligent Vehicles Symp.*, 2009.

[27] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2016.

[28] L. Nunes, R. Marcuzzi, X. Chen, J. Behley, and C. Stachniss. SegContrast: 3D Point Cloud Feature Representation Learning through Self-supervised Segment Discrimination. *IEEE Robotics and Automation Letters (RA-L)*, 7(2):2116–2123, 2022.

[29] L. Porzi, S.R. Bulo, A. Colovic, and P. Kontschieder. Seamless Scene Segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[30] R.B. Rusu and S. Cousins. 3D is here: Point Cloud Library (PCL). In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2011.

[31] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. In *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2014.

[32] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[33] H. Tang, Z. Liu, S. Zhao, Y. Lin, J. Lin, H. Wang, and S. Han. Searching Efficient 3D Architectures with Sparse Point-Voxel Convolution. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2020.

[34] A. Teichman, J. Levinson, and S. Thrun. Towards 3D object recognition via classification of arbitrary object tracks. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2011.

[35] D.Z. Wang, I. Posner, and P. Newman. What Could Move? Finding Cars, Pedestrians and Bicyclists in 3D Laser Data. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2012.

[36] K. Wong, S. Wang, M. Ren, M. Liang, and R. Urtasun. Identifying unknown instances for autonomous driving. In *Proc. of the Conf. on Robot Learning (CoRL)*, 2020.

[37] S. Xie, J. Gu, D. Guo, C.R. Qi, L. Guibas, and O. Litany. PointContrast: Unsupervised Pre-training for 3D Point Cloud Understanding. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2020.

[38] B. Yang, M. Liang, and R. Urtasun. Hdnet: Exploiting hd maps for 3d object detection. In *Proc. of the Conf. on Robot Learning (CoRL)*, 2018.

[39] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2021.

[40] M.D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2014.

[41] Z. Zhang, R. Girdhar, A. Joulin, and I. Misra. Self-Supervised Pre-training of 3D Features on any Point-Cloud. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2021.

[42] Z. Zhou, Y. Zhang, and H. Foroosh. Panoptic-PolarNet: Proposal-Free LiDAR Point Cloud Panoptic Segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[43] X. Zhu, H. Zhou, T. Wang, F. Hong, Y. Ma, W. Li, H. Li, and D. Lin. Cylindrical and Asymmetrical 3D Convolution Networks for LiDAR Segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.