Domain generalization of 3D semantic segmentation in autonomous driving

Jules Sanchez[1]   Jean-Emmanuel Deschaud[1]   François Goulette[1,2]

[1]*Centre for Robotics, Mines Paris - PSL,*
PSL University,
75006 Paris, France

firstname.surname@minesparis.psl.eu

[2]*U2IS, ENSTA Paris,*
Institut Polytechnique de Paris,
91120 Palaiseau, France

firstname.surname@ensta-paris.fr

Abstract

Using deep learning, 3D autonomous driving semantic segmentation has become a well-studied subject, with methods that can reach very high performance. Nonetheless, because of the limited size of the training datasets, these models cannot see every type of object and scene found in real-world applications. The ability to be reliable in these various unknown environments is called domain generalization.

Despite its importance, domain generalization is relatively unexplored in the case of 3D autonomous driving semantic segmentation. To fill this gap, this paper presents the first benchmark for this application by testing state-of-the-art methods and discussing the difficulty of tackling Laser Imaging Detection and Ranging (LiDAR) domain shifts.

We also propose the first method designed to address this domain generalization, which we call 3DLabelProp. This method relies on leveraging the geometry and sequentiality of the LiDAR data to enhance its generalization performances by working on partially accumulated point clouds. It reaches a mean Intersection over Union (mIoU) of 50.4% on SemanticPOSS and of 55.2% on PandaSet solid-state LiDAR while being trained only on SemanticKITTI, making it the state-of-the-art method for generalization (+5% and +33% better, respectively, than the second best method).

The code for this method is available on GitHub:
https://github.com/JulesSanchez/3DLabelProp.

1. Introduction

Since the release of SemanticKITTI [2], 3D semantic segmentation for autonomous driving has been a field of growing interest, resulting in the emergence of a large variety of open-source datasets [3, 27, 21] and high-performing methods [47, 29].

Using LiDAR data instead of the typical 2D images obtained from cameras has several advantages. First, LiDAR data are resilient to some sources of 2D domain shifts, such
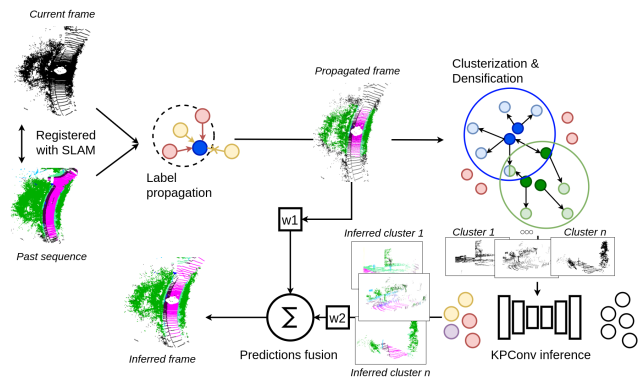

Figure 1: Illustration of the 3DLabelProp algorithm. With the past inferred sequence and the current registered LiDAR frame given as inputs, 3DLabelProp geometrically propagates the static object labels. Then, it clusterizes the current scan and populates it with points from the past sequence to feed small but dense point clouds to the KPConv module. Finally, predictions from the geometric propagation and deep learning module are combined to obtain the inferred LiDAR frame.

as the change in illumination. Furthermore, these data yield precise geometric information and reliable distance information relative to the objects in scenes (down to a few centimeters in error). Nevertheless, deep learning methods dealing with LiDAR data are still sensitive to domain shifts such as scene-type change (urban vs. suburban) or sensor change [17]. This observation is especially relevant for autonomous driving semantic segmentation because the source of the domain shift varies and originates from hardware, such as the number of fibers in the LiDAR and its positioning on the vehicles; from firmware such as reflectivity calculations; and from the scene itself. LiDAR data are also costly to acquire and label (1,700 hours of labelization for SemanticKITTI [2]). The accumulation of these issues has clarified the need for domain adaptation and domain generalization methods to, on the one hand leverage

the availability of synthetic data [8, 36] and, on the other hand, perform well on unseen data because it is not feasible to cover every scene type and corner case.

Data adaptation aims to bridge the domain shift with a learning transferable model; hence, data adaptation-methods usually train a model on source data and then perform an adaptation step, for which they have access to either sparsely labeled or fully unlabeled target data. Conversely, domain generalization methods work in a setup where the target data are fully unavailable.

Although domain adaptation has gained a lot of traction in 3D scene understanding, domain generalization is still relatively unexplored. Furthermore, the disparity between the label sets of various available datasets makes it difficult to compare methods across datasets. In this work, we propose a simple method to evaluate domain generalization performance. With it, we benchmark current state-of-the-art approaches for autonomous driving semantic segmentation.

Furthermore, we propose a novel method to perform semantic segmentation while improving generalization performances. Specifically, we develop a domain alignment method, which accumulates the sequences of LiDAR frames rather than examining scans individually, as is usually done. Although working on such point clouds is slow, we propose several geometry-based mechanisms to speed up the method by propagating the labels of static objects to new points and processing small clusters of points rather than the full point cloud. This method is called *3DLabel-Prop*.

We can summarize our contributions as follows:

- We properly introduce the domain generalization problem of 3D semantic segmentation in the autonomous driving field and evaluate current state-of-the-art methods.

- We propose a domain generalization-oriented method for autonomous driving semantic segmentation called 3DLabelProp (Figure 1).

2. Related works

2.1. Autonomous driving 3D semantic segmentation

To tackle the task of 3D autonomous driving semantic segmentation and its speed requirements, older permutations invariant methods were omitted [30, 40] in favor of methods reordering point clouds into structured representations. These methods are divided in two main approaches: projecting the point clouds in 2D with a range-based image [25][37] and bird's-eye-view image [44] or leveraging sparse convolution-based architectures as introduced by MinkowskiNet [6], under the name Sparse Residual U-Net (SRUNet). SRUNets have been used as the backbone for

more refined methods [6, 29, 47, 39], which are the best-performing methods on the various benchmarks. Some multimodal methods linking images and LiDAR are also starting to appear [42] but are outside the scope of the present article.

Because of the computation time constraint, most of these methods process frames one by one, discarding the sequentiality of the autonomous driving data. Recent works have started to exploit this for moving object segmentation [24] or for full segmentation thanks to flow estimation [22], 4D representations [23, 9], or 4D partial aggregation [1, 16].

Aggregation-based methods are similar to offline outdoor semantic segmentation, which is a well-studied field reaching great performances [31]. Nevertheless, these methods are designed for offline processing and are inherently slow and unsuited for autonomous driving segmentation as such.

3DLabelProp is inspired by aggregation-based methods, with a focus on improving speed and generalization performances.

2.2. Domain generalization for autonomous driving semantic segmentation

Domain generalization in 2D has been thoroughly explored, working on training techniques such as meta-learning [19], episodic training [20], and multitask learning [4]; data augmentation [32, 41]; or architecture design [26, 14, 15].

Meta-learning methods provide a different training algorithm rather than the typical approach, specifically in [19], which simulates training and testing domains at each batch and ensures that the training and testing gradients are aligned. Episodic training [20] pretrains several domain-specific feature extractors and then combines them at training time to make them robust against domain shifts. These methods rely on the availability of various source domains or the ability to simulate them at training time. Multitask learning aims to associate self-supervised or unsupervised tasks alongside the main one to learn better representations to be less sensitive to domain shifts. Data augmentation techniques provide a set of rules and methods to enhance the data at training time to diversify the available data [11], usually by modifying appearance and sensor information such as texture, brightness, and contrast [41]; these techniques can also specialize in finding relevant and difficult samples of the available data [32]. Finally, architecture design methods proposed new neural blocks to improve domain generalization, such as normalization blocks, to align samples from different domains [26].

A more thorough overview and discussions of the various domain generalization paradigms and algorithms can be found in [46] [34].

In point cloud processing, domain generalization has

been mainly evaluated and studied for 3D registration [38] [13] and object classification [10, 33]. For scene understanding, fewer studies have been released and have focused on object detection, such as 3D-VField [18], which proposes new data augmentation techniques by applying point cloud deformation. Besides these works, common LiDAR scene understanding works do not measure the domain generalization abilities of the various architectures.

While domain adaptation is different from domain generalization, unsupervised domain adaptation (UDA) is a closely related research area and is relevant to include here. We can distinguish sim-to-real UDA, which pretrains a model on synthetic data and adapts it on real data. UDA methods try to align the style of synthetic to real point clouds such as in SqueezeSeg [35] and ePointDA [45], even training GANs to learn and transfer scarcity and scene type [36]; there is also real-to-real domain adaptation, which employs alignment techniques such as learning completion on point clouds to train on this intermediate representations [43] or resampling point clouds to align the target data with the source data [17].

Similar to [43], 3DLabelProp proposes finding a common representation of autonomous driving LiDAR data, namely the partially accumulated point clouds, such that the features of the target point clouds are similar to the features of the source point clouds.

3. Domain generalization

3.1. Definition

Domain generalization addresses the transferability of a model or an algorithm from one source to a target domain, without the availability of the target domain prior to the inference. Nonetheless, this unavailability of the target domain is not synonymous with the absence of priors on the target data.

For autonomous driving LiDAR semantic segmentation, it can be assumed that point clouds will represent similar scenes, namely driving environments, hence representing similar objects, such as cars, pedestrians, and roads.

Consequently, we can identify three main domain shift components:

- Sensor shift, which stems from the resolution, setup, technology and model of the LiDAR sensor on the acquisition platform.

- Appearance shift, which originated from the different appearances of similar objects in different scenes, such as car sizes or vegetation variety. It also encapsulates weather and season change that can affect appearance of the objects such as tree losing their leaves in the winter, or fog adding noise to the acquisition.

- Scene shift, which is caused by different scenes displaying different objects in their environments; for example traffic lights are typical of an urban scene but not highway scenes.

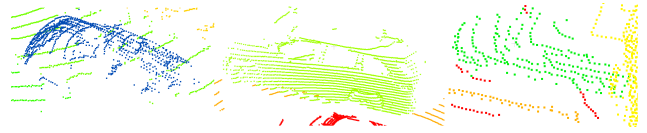Examples of appearance and sensor shifts can be seen in Figure 2.

Figure 2: Example of different cars. From left to right: in SemanticKITTI, in nuScenes, and in SemanticPOSS.

3.2. Evaluation

3.2.1 Datasets

name	LiDAR technology	# fibers
SemanticKITTI	rotative	64
SemanticKITTI32	rotative	32
Panda64	rotative	64
PandaFF	solid-state	150
SemanticPOSS	rotative	40
nuScenes	rotative	32

Table 1: Sensor information for the used datasets.

In order to benchmark the generalization performances of the state-of-the-art LiDAR semantic segmentation algorithms, we leverage several open-source datasets: SemanticKITTI [2], SemanticPOSS [27], nuScenes [3], and PandaSet [12].

SemanticKITTI is a standard, widely used dataset in 3D Vision and is based on a 360-degree rotating LiDAR with 64 fibers. We will use it as our main training source dataset. To study sensor shift in a configuration without any other shifts, we introduce SemanticKITTI32, which is generated by removing one out of two fibers from the original SemanticKITTI dataset.

We divide PandaSet into two sub-datasets, Panda64 (P64), which are the frames acquired by the 64-fiber LiDAR inside PandaSet, and PandaFF (PFF), which are the frames acquired by the front-facing LiDAR inside PandaSet. It is important to note that PandaFF is a dataset acquired by a solid-state LiDAR.

SemanticPOSS is a dataset acquired at a Chinese university campus by a 360-degree rotating LiDAR with 40 fibers. While it has a lower number of fibers than SemanticKITTI, they have a similar vertical angular resolution ($0.4°$ for SemanticKITTI and $0.33°$ for SemanticPOSS).

Due to the characteristics of the sensor of Panda64 and SemanticPOSS, we can assume that the sensor shift is min-

Method	Input type	$\text{mIoU}^{SK}_{\mathcal{L}_{SK}}$	$\text{mIoU}^{SK32}_{\mathcal{L}_{SK}}$	$\text{mIoU}^{P64}_{\mathcal{L}_{SK\cap PS}}$	$\text{mIoU}^{PFF}_{\mathcal{L}_{SK\cap PS}}$	$\text{mIoU}^{SP}_{\mathcal{L}_{SK\cap SP}}$	$\text{mIoU}^{NS}_{\mathcal{L}_{SK\cap NS}}$
CENet [5]	Range-based	58.8	39.1	13.3	4.9	27.9	5.0
Helix4D [23]	Point-based & 4D	60.0	53.2	27.7	14.2	36.0	34.3
KPConv [31]	Point-based	58.3	52.7	32.7	21.1	39.1	**46.7**
SRUNet [6]	Voxel-based	58.6	54.0	**44.2**	**22.2**	45.3	42.7
SPVCNN [29]	Voxel & point-based	**62.3**	**57.4**	40.2	19.4	**45.4**	45.1
C3D [47]	Cylindrical voxel-based	60.7	53.1	18.4	6.5	41.9	32.7

Table 2: Generalization benchmark when trained on SemanticKITTI and tested on SemanticKITTI (SK), SemanticKITTI32 (SK32), Panda64 (P64), PandaFF (PFF), SemanticPOSS (PS), and nuScenes (NS).

imal, thus they are used to measure scene shift and appearance shift.

nuScenes is acquired by a 360-degree rotating LiDAR with 32 fibers in a urban context. It displays many dynamic objects alongside sensor shift, scene shift, and appearance shift.

A summary can be found in Table 1.

3.2.2 Mean intersection over union

The evaluation of the transferability of a model from one dataset to another is not straightforward because their label sets are all different. This difference in labelling, coming from the scene shift, and preferences of the authors, makes the comparison more difficult than mIoU computations on the same label set.

To tackle these labeling differences, we realign the label set thanks to a label remapping inspired by [28]. Using similar coarse labels, we group together the labels that do not have a one-to-one mapping from one dataset to another while keeping the labels that do. This coarse-and-fine label set is different for each pair of datasets to measure more precisely the quality of the scene understanding.

This results in the definition of several label sets, from which we detail the two most important: $\mathcal{L}_{SK\cap SP}$ = {person, rider, bike, car, ground, trunk, vegetation, traffic-sign, pole, building, fence} between SemanticKITTI (SK) and SemanticPOSS (SP) and $\mathcal{L}_{SK\cap NS}$ = {motorcycle, bicycle, person, road, sidewalk, other ground, manmade, vegetation, car, terrain} between SemanticKITTI (SK) and nuScenes (NS). Details of the mapping from the original label sets to these coarse-and-fine label sets can be found in the Supplementary Material.

In order to clarify which label sets were used in each table, we introduce a specific notation for the mIoU: $\text{mIoU}^{target}_{labelset}$. For instance, when we evaluate results on SemanticPOSS (SP) with the label set $\mathcal{L}_{SK\cap SP}$ while the source training set is SemanticKITTI (SK), we compute the $\text{mIoU}^{SP}_{\mathcal{L}_{SK\cap SP}}$.

Although these label sets are used for evaluation, they were not used for training purposes. Every method was trained on the regular label set of the source data.

When we evaluate source-to-source segmentation, which means evaluating a model on a validation set that presents

no shift from the training set, we compute the results on the original label set.

3.2.3 Methods

The methods for this benchmark were chosen based on three factors: availability of the source code and the pre-trained models, performances on the SemanticKITTI and nuScenes semantic segmentation benchmarks, and representation of the various LiDAR semantic segmentation paradigms (range based, voxel based, point based, and sequence based).

As such, we selected 6 different architectures: CENet [5], Helix4D [23], KPConv [31], SRUNet [6], SPVCNN [29], and Cylinder3D [47].

While this benchmark is not fully exhaustive, we believe it is complete enough to propose an overview of the current generalization performances.

All these architectures were retrained to omit the reflectivity channel, the details of the training parameters can be found in the Supplementary Material. They were selected by using the official implementations and proposed parameters. We removed the reflectivity channel to improve the generalization performance. Details of the proof can be found in subsection 3.4.

3.3. Generalization benchmark

Our proposed generalization benchmark can be found in Table 2. We show the mIoU over the 6 generalization cases and on the source-to-source case, here evaluated on Sequence 08 of SemanticKITTI. The details of the per-category IoUs can be found in the Supplementary Material. The most obvious observation that can be made is that high source-to-source segmentation performances do not translate automatically into high generalization performances.

As all methods are trained without reflectivity, they have to rely on geometry only for learning. As such the first experiment (SemanticKITTI to SemanticKITTI32) shows the robustness against the sparsity of the scene. It highlights a very large sensitivity of the sparsity for the range-based method (CENet). Other methods are more robust but are still displaying a significant drop in performance (between -5% and -7%).

Panda64 and SemanticPOSS help us measure robustness to scene and appearance shifts. First, we see that performances can change drastically from one dataset to another, especially for Cylinder3D which ranks third on SemanticPOSS, but fifth on Panda64. However, we observe that overall voxel-based methods (SRUNet and SPVCNN) are the most robust, and point-based (KPConv and Helix4D) are right behind.

To further investigate the robustness to sensor shift, we put ourselves in a more difficult case with PandaFF. As the LiDAR is a solid-state LiDAR, the scan topology differs from the typical scans found in the other datasets (only 60 degrees front facing vs 360 degrees for other datasets). Furthermore, the range of the scan is much bigger (150m vs 50m for SemanticKITTI), which can cause issues for hyperparameter selection. We observe that no architecture reaches satisfying results. Nonetheless, KPConv, SRUNet, and SPVCNN are much more robust than other methods. Coupled with the results on Panda64, which displays the exact same scene with a different sensor, we can see that KPConv displays a better robustness to sensors than SRUNet and SPVCNN as it drops less performance from one modality to another.

Finally, results on nuScenes re-highlight some trends observed previsously. We especially observe the robustness of the voxel-based methods and of KPConv to scene and sensor shifts. We see that the ranking of the methods on nuScenes cannot be predicted by only examining the rankings for SemanticPOSS and SemanticKITTI32, even if together they represent the same shifts as nuScenes.

To conclude, we can see that CENet is overall the least satisfying methods for generalization. Cylinder3D displays a large sensitivity to sensor shift, but can otherwise reach decent performances. Helix4D has more difficulties than KPConv against sensor and scene shifts. Finally, KPConv, SRUNet, and SPVCNN are the most robust methods.

It is interesting to note that while we observe some trends on the generalization abilities of some methods, none of them systematically outperform every other method.

3.4. Effect of reflectivity on generalization

As mentioned in the introduction, although reflectivity is a very efficient feature for source-to-source semantic segmentation, we believe it is detrimental to domain generalization and very sensitive to sensor and even firmware changes (e.g., Velodyne HDL32 firmware change[1]). We demonstrate this in Table 3. We have found a large improvement for generalization when the model does not use reflectivity. In return, we observe a decrease in source-to-source segmentation performances.

With-reflectivity models were retrieved from the official implementation of the models when they were avail-

able. Otherwise we retrained them. We also retrained all reflectivity-less models following the same data augmentation patterns, using the official repositories and their proposed hyperparameters. As mentioned before, details can be found in the Supplementary Material.

In order to account for the difference in the reflectivity range from SemanticKITTI to SemanticPOSS, the reflectivity was rescaled which is a first step of domain adaptation, to achieve better results, for the models with reflectivity. SemanticPOSS was used for the reflectivity analysis due to the similarity in scan topology with SemanticKITTI and because it follows a more similar reflectivity distribution.

	Reflectivity	mIoU$_{\mathcal{L}_{SK}}^{SK}$	mIoU$_{\mathcal{L}_{SK \cap SP}}^{SP}$
CENet†	yes	**61.4**	27.5
	no	58.8	**27.9**
Helix4D†	yes	**63.1**	35.0
	no	60.0	**36.0**
KPConv*	yes	**59.9**	33.1
	no	58.3	**39.1**
SRUNet*	yes	**61.9**	45.2
	no	58.6	**45.3**
SPVCNN†	yes	**63.8**	36.9
	no	62.3	**45.4**
C3D†	yes	**66.9**	33.8
	no	60.7	**41.9**

Table 3: Study of the effect of using reflectivity as input for the neural networks, using SemanticKITTI (SK) for training and SemanticPOSS (SP) for testing. † means that with-reflectivity trained models were extracted from the official repository, * means it was retrained.

While source-to-source performances should not be degraded when performing generalization, when building the benchmark, we wanted to focus on generalization performances and thus we wanted to put each method in a favorable light to show an upper bound on their naive generalization methods. This is why the benchmark was built without reflectivity.

4. Sequence-based semantic segmentation

4.1. Motivation

Although autonomous driving 3D semantic segmentation has obtained satisfying results, we highlighted before that domain generalization could be improved and that the segmentation models are sensitive to domain shifts. To counter this, we propose a new approach to semantic segmentation, called 3DLabelProp (Figure 1). It is a four-step method designed to achieve natively better resilience to domain shifts.

It works on accumulated point clouds to create a partial independence from the acquisition sensors and leverages the sequentiality of the data through its label propagation module.

---

[1]HDL32E Software Manual

The label propagation module geometrically propagates a part of the labels inferred in the past to the current and newly acquired point cloud. Because it is purely geometric, it is a very fast module with very high accuracy.

Because of the size of accumulated point clouds, it is tedious and slow to apply deep learning methods to it. To speed up the deep learning process while retaining the high-quality geometric information of the accumulated point cloud, we clusterize it to generate smaller point clouds. These small clusters can then be fed to a high-performance neural network while working at a decent speed thanks to the small size of these extracted point clouds. We chose KPConv as our deep learning model because of its high performance on accumulated point clouds and decent generalization performances.

We detail each block of 3DLabelProp in the following subsections.

4.2. Point cloud accumulation

To accumulate consecutive LiDAR frames, we apply the state-of-the-art CT-ICP LiDAR odometry [7] without loop closure. Once we have the relative pose, we align all the point clouds on the world coordinate. To process the current frame, we accumulate the last 20 frames with it and perform grid subsampling to reduce the memory footprint without altering the geometry. We also remove points too far from the center of the current frames because they add no geometric neighborhoods to the relevant points while having a high memory footprint.
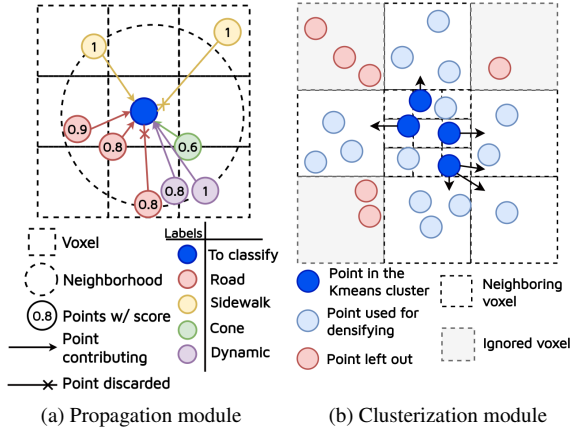

(a) Propagation module   (b) Clusterization module

Figure 3: Illustration of the geometric modules of 3DLabelProp.

4.3. Label propagation

Contrary to offline semantic segmentation, sequence-based semantic segmentation incorporates incremental information in the accumulated point cloud and retains past predictions. Although single scan methods completely discard it, in 3DLabelProp, we leverage this past information to propagate information geometrically in the newly registered scan.

Defining $L = (l_i)_{i \in \{0, K\}}$ as the semantic label set, we split it into two subsets, $D = (d_i)_{i \in \{0, K_d\}}$ and $S = (s_i)_{i \in \{K_d, K\}}$ dynamic and static labels, respectively, such that $L = D \cup S$. Static labels are defined as points that cannot move in the world coordinate, from one scan to another. On the contrary, a dynamic object *can* move from one scan to another. The ground and buildings are examples of static labels, whereas cars and pedestrians are examples of the dynamic labels.

As defined, objects that are part of static labels are expected not to move. Furthermore, two very close LiDAR points in the world coordinates can be assumed to be part of the same static object. As such, we can *propagate* past inferences to present points if they are close enough that they are expected to be part of the same object. To achieve this, we process it in a two-step pipeline. First, we gather neighborhoods for each point of the newly acquired scan, and then, we assign it a propagation score representing confidence in a geometrical prediction.

To extract neighbors, we assess a sphere centered on the current point processed and gathering all points belonging to the past accumulation inside. This neighbor search is sped up by the grid subsampling beforehand and the ability to leverage these pre-computed voxels. Then, based on the label and its associated score of each neighbor, we proceed to voting to extract the predicted label. Neighbors that have a distance-weighted confidence too low are discarded as they are considered unreliable.

If the majority vote is a dynamic label then we assign no value. Otherwise, we assign the majority label and the weighted average of the confidence of the neighbors belonging to this label as the new score. The process is illustrated in Figure 3.

We introduce $\mathcal{P}_{acc} \in \mathbb{R}^{N \times 3}$, the accumulated point clouds made of the past sequence; $Y_{acc} \in \{0, K\}^N$ its associated inferred labels, and $\mathcal{C}_{acc} = (c_{acc,i} \times e_{y_i})_{i \in \{1, N\}}$ the one-hot encoded confidence vector associated with each label, $c_{acc,i}$ the confidence score, and e_{y_i} the one-hot vector with the one at the y_ith position. Similarly, we introduce $\mathcal{P}_{curr} \in \mathbb{R}^{M \times 3}$, Y_{curr}, and $\mathcal{C}_{curr}$ which are the associated values for the current LiDAR frame. Because it is not yet processed, the labels are initialized at -1, and the associated confidence is zero. We then proceed to the propagation:

$$\forall p_{curr,i}, i \in \{1, M\};$$

$$c_{curr,i} = \sum_{p_{acc,n} \in \mathcal{N}(p_{curr,i})} w(i,n) \mathbb{1}_{w(i,n) > 0.5}; \quad (1)$$

Method	Input type	$\text{mIoU}^{SK}_{\mathcal{L}_{SK}}$	$\text{mIoU}^{SK32}_{\mathcal{L}_{SK}}$	$\text{mIoU}^{P64}_{\mathcal{L}_{SK\cap PS}}$	$\text{mIoU}^{PFF}_{\mathcal{L}_{SK\cap PS}}$	$\text{mIoU}^{SP}_{\mathcal{L}_{SK\cap SP}}$	$\text{mIoU}^{NS}_{\mathcal{L}_{SK\cap NS}}$
KPConv [31]	Point-based	58.3	52.7	32.7	21.1	39.1	**46.7**
SPVCNN [29]	Voxel & point-based	**62.3**	**57.4**	40.2	19.4	45.4	45.1
C3D [47]	Cylindrical voxel-based	60.7	53.1	18.4	6.5	41.9	32.7
3DLabelProp (Ours)	Point-based & 4D	60.8	56.7	**56.0**	**55.2**	**50.4**	44.4

Table 4: Generalization results when trained on SemanticKITTI and tested on SemanticKITTI (SK), SemanticKITTI32 (SK32), Panda64 (P64), PandaFF (PFF), SemanticPOSS (PS), and nuScenes (NS).

with $w(i,n) = d(p_{acc,n}, p_{curr,i}) \times c_{acc,i}$ and $d(p,q) = e^{-\frac{||p-q||^2}{\sigma^2}}$, where σ being the bandwidth hyperparameter, giving the range of the neighbor search. We add a cut-off function in which points with too low confidence relative to the position of the current points are discarded. $\mathcal{N}(p_{curr,i})$ is the neighborhood of $p_{curr,i}$. Because the past sequence $\mathcal{P}_{acc}$ is grid-subsampled to keep the memory consumption low, we reuse this voxelization to perform a radius-based neighbor search. Because we know the bandwidth of the label propagation, we can restrict our search to the voxels contained in this sphere, hence speeding up the process. Then we have the following:

$$y_{curr,i} = \begin{cases} \arg\max(c_{curr,i}), \text{ if } & \arg\max(c_{curr,i}) > K_d \\ -1, & \text{otherwise} \end{cases}$$
(2)

with $\arg\max(c_{curr,i}) > K_d$, meaning it is a static label (K_d is the number of dynamic labels).

When fed with past ground truth point clouds, this step reaches 95% accuracy over 80% of the static points. Only 15% of the dynamic points receives a wrongful prediction, making it a robust step.

4.4. Clusterization

To efficiently process a large point cloud with KPConv, it needs to be split into smaller point clouds. While the original authors created sub-point clouds by extracting spheres inside the original one, the number of spheres necessary to cover a whole accumulated point cloud is prominent, resulting in a large amount of overlap. To counter this, we propose using a clustering algorithm, covering the point cloud more efficiently and with minimal overlap.

Because we do not need to infer the whole accumulated point clouds but only the remaining points after the label propagation step, we apply the clustering algorithms only on $p_{curr,i}$, where $y_{curr,i} = -1$, which results in fewer points than the original point clouds and allows for a sophisticated clustering algorithm to run at a high speed. As such, we use the K-means algorithm to create N_c clusters.

This step creates very sparse clusters that are then completed with points from the accumulated point cloud to create dense neighborhoods for those few remaining uninferred points. In order to do this, we voxelize the accumulated point cloud, and we fetch each point that belongs to either a current voxel occupied by points of the cluster or points belonging to neighbors of a voxel occupied by points of the cluster.

Not all the voxels touching the current one are neighbor voxels. Rather, they are computed based on where inside the current voxel the points of the cluster are, precisely. To do this, we divide the voxel into 27 sub-voxels (3x3x3) and assign to each sub-voxel a specific neighborhood. For instance in 2D we divide each pixel in 9 sub-pixel (3x3), and for the middle-left pixel the only neighbor is the left-neighbor. The process is illustrated in 2D in Figure 3.

These completed clusters are the input of our KPConv module.

4.5. Deep learning

Finally, we feed the clusters to KPConv, which performs very efficiently on dense point clouds. Because the clusters are relatively small, we reach a higher inference speed than typical KPConv use cases. Because the clusters have a small overlap resulting from cluster completion, some points can undergo several inferences. To have the final predicted label, we average all confidence scores and take the highest value. This average score is the newly assigned confidence.

The resulting pipeline allows us to process accumulated point clouds at a higher pace than typical approaches, achieving very good generalization performance.

The details of the KPConv model used in our pipeline can be found in the supplementary materials.

4.6. Final prediction

The last step is to fuse predictions from the propagation module and the deep learning module as some points undergo predictions from both modules. In order to achieve this, we weight the predictions of the label propagation module by w1 and of the deep learning module by w2. This novel score is then used to infer the final label predicted. In pratice, as KPConv is much more reliable than the label propagation module we use w1 = 0 and w2 = 1.

In addition to reproducing the same experiments as the benchmark, we expanded it by training on nuScenes and testing on the same datasets in Table 5.

Method	Input type	mIoU$^{NS}_{\mathcal{L}_{NS}}$	mIoU$^{SK}_{\mathcal{L}_{NS\cap SK}}$	mIoU$^{SK32}_{\mathcal{L}_{NS\cap SK}}$	mIoU$^{P64}_{\mathcal{L}_{NS\cap PS}}$	mIoU$^{PFF}_{\mathcal{L}_{NS\cap PS}}$	mIoU$^{SP}_{\mathcal{L}_{NS\cap SP}}$
KPConv [31]	Point-based	63.1	44.9	50.6	25.0	16.9	60.7
SPVCNN [29]	Voxel & point-based	67.2	49.4	53.2	43.7	11.1	**64.8**
C3D [47]	Cylindrical voxel-based	70.2	31.7	46.1	15.8	4.7	42.8
3DLabelProp (Ours)	Point-based & 4D	**71.0**	**60.5**	**62.5**	**65.4**	**66.7**	64.3

Table 5: Generalization results when trained on nuScenes and tested on nuScenes (NS), SemanticKITTI (SK), SemanticKITTI32 (SK32), Panda64 (P64), PandaFF (PFF), and SemanticPOSS (PS).

5. Results

In this section, we discuss and study 3DLabelProp to understand the impact of the various implementation choices on the generalization performance. Rather than copying the full benchmark from subsection 3.3, we only write results from Cylinder3D, SPVCNN, and KPConv for comparison with our method. This choice was made as KPConv is the baseline for our model, Cylinder3D is the standing state-of-the-art method for source-to-source segmentation (when reflectivity is available), and SPVCNN showed the overall best generalization performances.

5.1. Domain generalization from SemanticKITTI

We detail the results from 3DLabelProp in Table 4. First, we observe that 3DLabelProp reaches satisfying source-to-source segmentation results, outperforming KPConv by more than 2%.

We built 3DLabelProp to decrease sensitivity to domain shift and precisely to sensor shift by working on the accumulated point cloud. 3DLabelProp displays the smallest decrease in performance from SemanticKITTI to SemanticKITTI32 (-4%) and between Panda64 and PandaFF (-0.8%). It is also the only method reaching satisfying performance on PandaFF (+33% from the second best).

Finally, on generalization performances, 3DLabelProp reaches the best performances for Panda64 (+12% from second best) and for SemanticPOSS (+5% from second best) and is the third best (-2.3%) on nuScenes. As we demonstrated the relatively low sensitivity from sensor shift for 3DLabelProp, nuScenes' results stem from scene shift. In particular, when studying per-class results in the Supplementary Material, we observe that the performances of KPConv on pedestrians explain its results.

Overall, 3DLabelProp is the state of the art for generalization regarding robustness and performance.

5.2. Domain generalization from nuScenes

For the state-of-the-art methods, we see similar patterns when trained with nuScenes and trained with SemanticKITTI. We note the sensitivity to domain shifts, as methods systematically perform better on SemanticKITTI32 than SemanticKITTI.

For source-to-source segmentation, we see that 3DLabelProp reaches state-of-the-art performance, showing that working on a denser representation is helpful, especially for low resolution LiDARs. Furthermore, we see patterns of results similar to results from Table 4. We see relatively low sensitivity to sensor shift as there is a very limited performance change from SemanticKITTI32 to SemanticKITTI (-2%), and the lowest change in performance between Panda64 and PandaFF (-1%). It is also on par with SPVCNN for SemanticPOSS (-0.5%) and outperforming it for Panda64 (+21.7%).

5.3. Related work comparison

Similarly to Complete & Label (C&L) [43], we believe that a dense representation of point clouds could reduce the domain gap. The way to achieve this representation is different. We accumulate our point clouds based on geometrical information, whereas they rely on learning of a completion network. We ensure the preservation of geometric properties scenes without additional hyperparameters. We compare our generalization results with theirs in Table 6. Furthermore, we compute the relative decrease from source-to-source to generalization to highlight the effectiveness of our method compared to theirs (which is Unsupervised Domain Adaptation, UDA, training using target data information without target labels, whereas we use no information about the target set). We are much more accurate from source-to-source and are either on par or quite better regarding relative decrease. We believe it stems from their difficulty in effectively training a completion model that generalizes well, whereas we use only the geometry for domain alignment.

	SK → NS			NS → SK		
Method	SK	NS	% drop	NS	SK	% drop
C&L [43]	50.2	31.6	-37%	54.4	33.7	-38%
3DLabelProp	69.0	42.7	-38%	66.5	50.5	-24%

Table 6: Comparison with C&L. All results were computed on the C&L label set.

5.4. Ablation study

In the ablation study, we examined the impact of several design choices in the conception of 3DLabelProp. All variants (including 3DLabelProp) use the same model design, which is the one provided by the authors of KPConv. Training parameters differ for 3DLabelProp compared to

	$\text{mIoU}^{SK}_{\mathcal{L}_{SK}}$	$\text{mIoU}^{SP}_{\mathcal{L}_{SK\cap SP}}$	$\text{mIoU}^{NS}_{\mathcal{L}_{SK\cap NS}}$	FPS
KPConv w/ reflectivity	59.9	33.1	47.6	0.2
KPConv	58.3	39.1	46.7	0.2
KPConv multiframe	53.0	47.2	44.2	0.05
3DLabelProp (Ours)	60.8	50.4	44.4	1.0

Table 7: Ablation study on 3DLabelProp

d_{prop}	N_c	N_f	$\text{mIoU}^{NS}_{\mathcal{L}_{NS}}$	$\text{mIoU}^{SK}_{\mathcal{L}_{NS\cap SK}}$	FPS
0.30m	20	20	71.0	60.5	1.2
0.10m	20	20	72.4	60.2	0.6
0.60m	20	20	69.1	57.2	1.5
0.30m	5	20	71.3	61.5	1.3
0.30m	40	20	68.8	59.7	0.9
0.30m	20	5	67.4	60.0	1.5
0.30m	20	10	70.9	60.2	1.5
0.30m	20	40	67.9	59.5	1.0

Table 8: Study of the geometric parameters. The first line is the model used in the paper.

the standard KPConv, due to the format of the input point cloud, which is a sphere for KPConv, and a cluster of unknown shape for 3DLabelProp. For KPConv multiframe we used the past 10 scans for accumulation.

First, we again note the results of KPConv with or without the reflectivity channel. Dropping the reflectivity channel triggers a huge gain in generalization from SemanticKITTI to SemanticPOSS for a very small loss from SemanticKITTI to nuScenes. The second step is computing KPConv on an accumulated frame, which also helps improve the generalization performances. Accumulating frames in SemanticPOSS and SemanticKITTI accentuates their geometric similarity, which results in extremely dense point clouds.

Using the full 3DLabelProp has two advantages: it keeps on improving the overall generalization performance when compared with other KPConv approaches and the various geometric modules help speed up the inference to better reach decent performances. The speed is computed for the inference on nuScenes.

5.5. Geometric parameters

For the propagation and clusterization steps of 3DLabelProp, we introduced 5 parameters: v_s, size of the voxels for the grid subsampling; d_{prop}, the distance of the propagation, which is linked to the bandwidth parameter σ; N_c, the number of clusters; V_c, the size of the voxels for the cluster densification; and N_f the number for frames used in the accumulated point cloud. The relationship between d_{prop} and σ, the bandwith parameter, is: $\sigma = \frac{d_{prop}}{\sqrt{ln(2)}}$

For all the trainings and all the inferences, the parameters chosen were $v_s = 0.05$m, $d_{prop} = 0.30$m, $V_c = 2$m, and $N_c = 10$ when the training data is SemanticKITTI, and $N_c = 20$ when the training data is nuScenes. Only N_c changes depending on the training model. We have a larger N_c for nuScenes as the training data is sparser and we want to create more overlap between clusters.

It is important to note that to respect the principle of generalization, these parameters are chosen on the training set and are not modified for all the inferences on all the generalization datasets. The two most important parameters of this geometric part are d_{prop} for the propagation distance and N_c for the number of clusters. We did an ab-

lation study of these parameters in Table 8 by training on nuScenes and testing the generalization on SemanticKITTI. We chose nuScenes as the training data due to its smaller training time.

First, we observe that even with ill-selected parameters ($d_{prop}=0.60$m), 3DLabelProp still achieves state-of-the-art generalization from nuScenes to SemanticKITTI. Then, we see the robustness, performance-wise, to the parameter selection, as there is only a difference of 4% between the best and worst performances.

The largest impact is on the process time. We see that d_{prop} is the main source of speed variation. We selected $d_{prop}=0.30$m as a trade-off between speed and accuracy, as the increase in performance between 0.30m and 0.10m is minimal but the speed drop is quite large.

The cluster parameter is more linked to memory consumption. The larger the number of clusters, the less memory consumption, up until a certain point where too many clusters create a lot of overlap and re-increase the memory consumption. It is not reported here as it has a small impact on nuScenes, but can cause issue in denser point cloud such as SemanticKITTI, which explains our choice of $N_c=20$ rather than $N_c=5$.

6. Conclusion

We have proposed the first 3D autonomous driving semantic segmentation domain generalization benchmark, highlighting the necessary work to be done in this area. We specifically highlighted how overfitting models on source-to-source segmentation is detrimental to domain generalization performance, whether done through architecture design or feature vector selection.

To fill the gap in the literature, we have proposed the first method designed for this task. Using two different source datasets and over 5 other target datasets we demonstrated 3DLabelProp is the most balanced and accurate generalization method.

References

[1] Mehmet Aygun, Aljosa Osep, Mark Weber, Maxim Maximov, Cyrill Stachniss, Jens Behley, and Laura Leal-Taixe.

4d panoptic lidar segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5527–5537, June 2021. 2

[2] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jürgen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9296–9306, 2019. 1, 3

[3] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11618–11628, 2020. 1, 3

[4] Fabio M. Carlucci, Antonio D'Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. Domain generalization by solving jigsaw puzzles. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2224–2233, 2019. 2

[5] Hui–Xian Cheng, Xian–Feng Han, and Guo–Qiang Xiao. Cenet: Toward concise and efficient lidar semantic segmentation for autonomous driving. In *2022 IEEE International Conference on Multimedia and Expo (ICME)*, pages 01–06, 2022. 4

[6] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3070–3079, 2019. 2, 4

[7] Pierre Dellenbach, Jean-Emmanuel Deschaud, Bastien Jacquet, and François Goulette. Ct-icp: Real-time elastic lidar odometry with loop closure. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 5580–5586, 2022. 6

[8] Jean-Emmanuel Deschaud. KITTI-CARLA: a KITTI-like dataset generated by CARLA Simulator. *arXiv e-prints*, 2021. 2

[9] Hehe Fan, Xin Yu, Yuhang Ding, Yi Yang, and Mohan Kankanhalli. Pstnet: Point spatio-temporal convolution on point cloud sequences. In *International Conference on Learning Representations*, 2021. 2

[10] H. Huang C. Chen Y. Fang. Manifold adversarial learning for cross-domain 3d shape representation. In *ECCV 2022*, 2022. 3

[11] D. Hendrycks, S. Basart, N. Mu, S. Kadavath, F. Wang, E. Dorundo, R. Desai, T. Zhu, S. Parajuli, M. Guo, D. Song, J. Steinhardt, and J. Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8320–8329, Los Alamitos, CA, USA, oct 2021. IEEE Computer Society. 2

[12] Scale.AI Hesai. Pandaset open datasets. https://scale.com/open-datasets/pandaset, 2020. [Online; accessed 18-February-2022]. 3

[13] S. Horache, J. Deschaud, and F. Goulette. 3d point cloud registration with multi-scale architecture and unsupervised transfer learning. In *2021 International Conference on 3D Vision (3DV)*, pages 1351–1361, Los Alamitos, CA, USA, dec 2021. IEEE Computer Society. 3

[14] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1510–1519, 2017. 2

[15] Xin Jin, Cuiling Lan, Wenjun Zeng, and Zhibo Chen. Style normalization and restitution for domain generalization and adaptation. *IEEE Transactions on Multimedia*, 24:3636–3651, 2022. 2

[16] Lars Kreuzberg, Idil Esen Zulfikar, Sabarinath Mahadevan, Francis Engelmann, and Bastian Leibe. 4d-stop: Panoptic segmentation of 4d lidar using spatio-temporal object proposal generation and aggregation. In *European Conference on Computer Vision Workshop*, 2022. 2

[17] Ferdinand Langer, Andres Milioto, Alexandre Haag, Jens Behley, and Cyrill Stachniss. Domain transfer for semantic segmentation of lidar data using deep neural networks. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8263–8270, 2020. 1, 3

[18] A. Lehner, S. Gasperini, A. Marcos-Ramiro, M. Schmidt, M. Mahani, N. Navab, B. Busam, and F. Tombari. 3d-vfield: Adversarial augmentation of point clouds for domain generalization in 3d object detection. In *CVPR*, pages 17274–17283, 2022. 3

[19] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy Hospedales. Learning to generalize: Meta-learning for domain generalization. In *AAAI Conference on Artificial Intelligence*, 2018. 2

[20] Da Li, Jianshu Zhang, Yongxin Yang, Cong Liu, Yi-Zhe Song, and Timothy Hospedales. Episodic training for domain generalization. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1446–1455, 2019. 2

[21] Yiyi Liao, Jun Xie, and Andreas Geiger. KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *Pattern Analysis and Machine Intelligence (PAMI)*, 2022. 1

[22] Xingyu Liu, Mengyuan Yan, and Jeannette Bohg. Meteornet: Deep learning on dynamic 3d point cloud sequences. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9245–9254, 2019. 2

[23] Romain Loiseau, Mathieu Aubry, and Loic Landrieu. Online segmentation of lidar sequences: Dataset and algorithm. *ECCV*, 2022. 2, 4

[24] Benedikt Mersch, Xieyuanli Chen, Ignacio Vizzo, Lucas Nunes, Jens Behley, and Cyrill Stachniss. Receding moving object segmentation in 3d lidar data using sparse 4d convolutions. *IEEE Robotics and Automation Letters*, 7(3):7503–7510, 2022. 2

[25] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. Rangenet ++: Fast and accurate lidar semantic segmentation. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4213–4220, 2019. 2

[26] Xingang Pan, Ping Luo, Jianping Shi, and Xiaoou Tang. Two at once: Enhancing learning and generalization capacities via ibn-net. In *ECCV*, 2018. 2

[27] Yancheng Pan, Biao Gao, Jilin Mei, Sibo Geng, Chengkun Li, and Huijing Zhao. Semanticposs: A point cloud dataset with large quantity of dynamic instances. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 687–693, 2020. 1, 3

[28] Jules Sanchez, Jean-Emmanuel Deschaud, and François Goulette. Cola: Coarse label pre-training for 3d semantic segmentation of sparse lidar datasets, 2022. 4

[29] Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3d architectures with sparse point-voxel convolution. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 685–702, Cham, 2020. Springer International Publishing. 1, 2, 4, 7, 8

[30] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. Tangent convolutions for dense prediction in 3d. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3887–3896, 2018. 2

[31] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6410–6419, 2019. 2, 4, 7, 8

[32] Riccardo Volpi, Hongseok Namkoong, Ozan Sener, John C Duchi, Vittorio Murino, and Silvio Savarese. Generalizing to unseen domains via adversarial data augmentation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. 2

[33] H .Chao C. Zhangjie Y. Wang J. Wang and M. Long. Metasets: Meta-learning on point sets for generalizable representations. In *2021 CVPR*, 2021. 3

[34] Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Tao Qin, Wang Lu, Yiqiang Chen, Wenjun Zeng, and Philip Yu. Generalizing to unseen domains: A survey on domain generalization. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2022. 2

[35] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. *ICRA*, 2018. 3

[36] Aoran Xiao, Jiaxing Huang, Dayan Guan, Fangneng Zhan, and Shijian Lu. Transfer learning from synthetic to real lidar point cloud for semantic segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 2795–2803, 2022. 2, 3

[37] Chenfeng Xu, Bichen Wu, Zining Wang, Wei Zhan, Peter Vajda, Kurt Keutzer, and Masayoshi Tomizuka. Squeeze-segv3: Spatially-adaptive convolution for efficient point-cloud segmentation. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 1–19, Cham, 2020. Springer International Publishing. 2

[38] Jiabo Xu, Yukun Huang, Zeyun Wan, and Jingbo Wei. Glorn: Strong generalization fully convolutional network for low-overlap point cloud registration. *IEEE Transactions on Geoscience and Remote Sensing*, pages 1–1, 2022. 3

[39] Jianyun Xu, Ruixiang Zhang, Jian Dou, Yushi Zhu, Jie Sun, and Shiliang Pu. Rpvnet: A deep and efficient range-point-voxel fusion network for lidar point cloud segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16024–16033, October 2021. 2

[40] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, pages 90–105, Cham, 2018. Springer International Publishing. 2

[41] Z. Xu, D. Liu, J. Yang, C. Raffel, and M. Niethammer. Robust and generalizable visual representation learning via random convolutions. In *ICLR*, 2021. 2

[42] Xu Yan, Jiantao Gao, Chaoda Zheng, Chao Zheng, Ruimao Zhang, Shuguang Cui, and Zhen Li. 2dpass: 2d priors assisted semantic segmentation on lidar point clouds. In *ECCV*, 2022. 2

[43] L. Yi, B. Gong, and T. Funkhouser. Complete & label: A domain adaptation approach to semantic segmentation of lidar point clouds. In *CVPR*, 2021. 3, 8

[44] Yang Zhang, Zixiang Zhou, Philip David, Xiangyu Yue, Zerong Xi, Boqing Gong, and Hassan Foroosh. Polarnet: An improved grid representation for online lidar point clouds semantic segmentation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9598–9607, 2020. 2

[45] Sicheng Zhao, Yezhen Wang, Bo Li, Bichen Wu, Yang Gao, Pengfei Xu, Trevor Darrell, and Kurt Keutzer. epointda: An end-to-end simulation-to-real domain adaptation framework for lidar point cloud segmentation. In *AAAI*, 2021. 3

[46] Kaiyang Zhou, Ziwei Liu, Yu Qiao, Tao Xiang, and Chen Change Loy. Domain generalization: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–20, 2022. 2

[47] Xinge Zhu, Hui Zhou, Tai Wang, Fangzhou Hong, Yuexin Ma, Wei Li, Hongsheng Li, and Dahua Lin. Cylindrical and asymmetrical 3d convolution networks for lidar segmentation. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9934–9943, 2021. 1, 2, 4, 7, 8