HAL open science

Image-to-Lidar Self-Supervised Distillation for Autonomous Driving Data

Corentin Sautier, Gilles Puy, Spyros Gidaris, Alexandre Boulch, Andrei Bursuc, Renaud Marlet

To cite this version:

Corentin Sautier, Gilles Puy, Spyros Gidaris, Alexandre Boulch, Andrei Bursuc, et al.. Image-to-Lidar Self-Supervised Distillation for Autonomous Driving Data. Conference on Computer Vision and Pattern Recognition, IEEE/CVF, Jun 2022, New Orleans, United States. pp.9881-9891, 10.1109/CVPR52688.2022.00966 . hal-04158465

HAL Id: hal-04158465

https://hal.science/hal-04158465v1

Submitted on 11 Jul 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Image-to-Lidar Self-Supervised Distillation for Autonomous Driving Data

Corentin Sautier [1], Gilles Puy[1], Spyros Gidaris[1], Alexandre Boulch[1], Andrei Bursuc[1], Renaud Marlet[1,2]
[1]valeo.ai, Paris, France
[2]LIGM, Ecole des Ponts, Univ. Gustave Eiffel, CNRS, Marne-la-Vallée, France

Abstract

Segmenting or detecting objects in sparse Lidar point clouds are two important tasks in autonomous driving to allow a vehicle to act safely in its 3D environment. The best performing methods in 3D semantic segmentation or object detection rely on a large amount of annotated data. Yet annotating 3D Lidar data for these tasks is tedious and costly. In this context, we propose a self-supervised pre-training method for 3D perception models that is tailored to autonomous driving data. Specifically, we leverage the availability of synchronized and calibrated image and Lidar sensors in autonomous driving setups for distilling self-supervised pre-trained image representations into 3D models. Hence, our method does not require any point cloud nor image annotations. The key ingredient of our method is the use of superpixels which are used to pool 3D point features and 2D pixel features in visually similar regions. We then train a 3D network on the self-supervised task of matching these pooled point features with the corresponding pooled image pixel features. The advantages of contrasting regions obtained by superpixels are that: (1) grouping together pixels and points of visually coherent regions leads to a more meaningful contrastive task that produces features well adapted to 3D semantic segmentation and 3D object detection; (2) all the different regions have the same weight in the contrastive loss regardless of the number of 3D points sampled in these regions; (3) it mitigates the noise produced by incorrect matching of points and pixels due to occlusions between the different sensors. Extensive experiments on autonomous driving datasets demonstrate the ability of our image-to-Lidar distillation strategy to produce 3D representations that transfer well on semantic segmentation and object detection tasks.

1. Introduction

Lidar sensors deliver rich information about the 3D world, and making sense of this kind of information is crucial for an autonomous driving vehicle to properly act in its environment, across different external conditions. State-of-the-art methods for semantic segmentation or object detection in Lidar point clouds rely on deep neural networks trained on large collections of annotated point clouds. Yet, annotating 3D Lidar point clouds is a long and costly task [3, 20]. Self-supervision reduces the burden of annotating large datasets by exploiting a large amount of non-annotated data to pre-train neural networks, which are subsequently fine-tuned on a smaller set of annotated data.

The current best performing self-supervised techniques for 3D neural networks working on real point clouds are mostly adapted to indoor scenes with dense point clouds. These methods suffer from several shortcomings when dealing with sparse point clouds, such as those acquired outdoor by a moving vehicle. For example, PointContrast [67] requires pairs of registered point clouds and a list of matching points between them. While multiple reliable matching points can be found in a densely sampled static scene, the number of such pairs of points is much lower in autonomous driving datasets, in particular on objects of interest (cars, trucks, pedestrians, etc.) as they are sparsely sampled and likely to move between two acquisitions. DepthContrast [72] avoids the need of finding pairs of corresponding points as it only requires a single representation for each scene. This representation is computed by global pooling and therefore loses information on small objects. These design choices limit significantly the performance of these methods in our experiments on autonomous driving scenes.

Our goal is to design a self-supervised method for tasks such as semantic segmentation or object detection in Lidar point clouds, and tailored to autonomous driving data. Most autonomous driving vehicles are equipped with an array of cameras and Lidar sensors that are synchronized and calibrated, offering rich surround-view information. These data are a lot easier to acquire than to annotate, and we propose to leverage them to distill self-supervised pre-trained image representations into a 3D network. This whole pre-training process does not require any annotation of the images nor of the point clouds. Self-supervised pre-training on images has proven very successful for learning generic representations

Code available at https://github.com/valeoai/SLidR

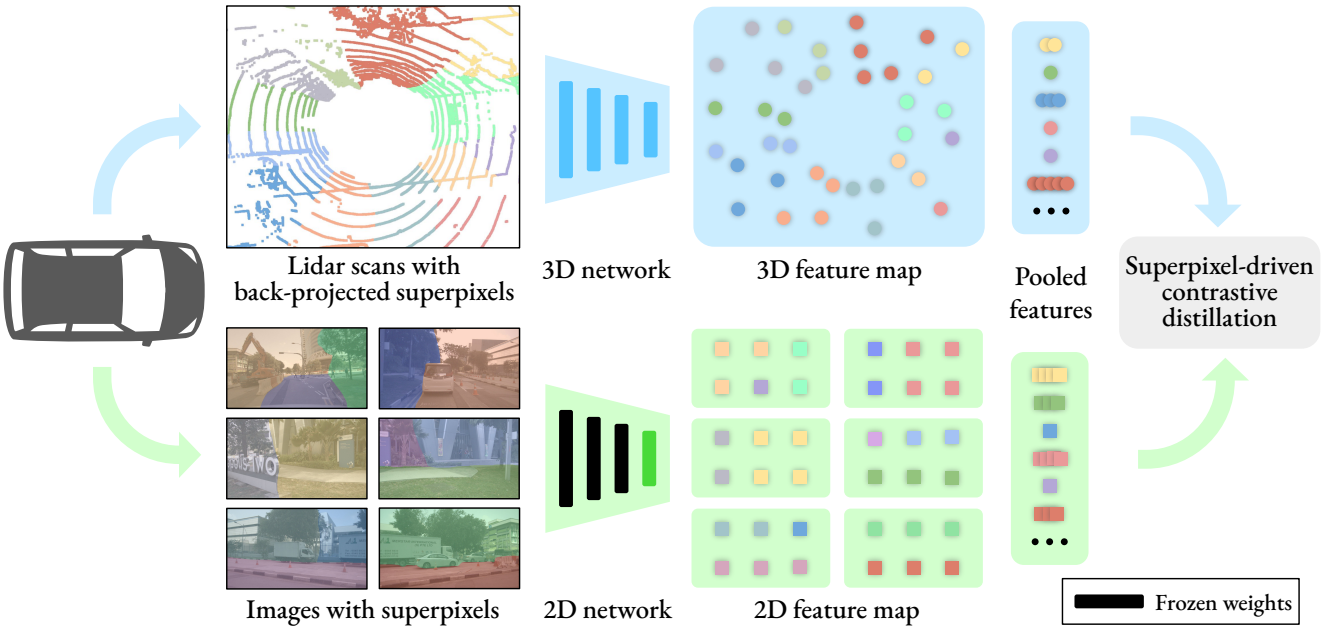

Figure 1. SLidR distillates the knowledge of a pre-trained and fixed 2D network into a 3D network. It uses superpixels to pool features of visually similar regions together, both on the images, and on the point clouds through superpixels back-projection. The superpixel-driven contrastive loss aligns the pooled point and image features. The visualized segments proposed in this figure have been manually generated and are intentionally over-sized for illustrative purposes. Superpixels actually used can be observed on Fig. 2.

that transfer well to various complex downstream tasks in 2D, often surpassing supervised pre-training [8, 22, 24, 28]. In this work, we show that these powerful representations can also be used to pre-train 3D networks for autonomous driving. We call this setting self-supervised 2D-to-3D representation distillation.

We propose a distillation loss suited to tasks such as semantic segmentation and object detection by forcing the networks to produce object-aware representations. Inspired by [31], we use superpixels [1, 18] which group visually similar regions that are likely to belong to the same object. We then use these superpixels as pooling masks for 3D point features and 2D pixel features, and enforce pairs of corresponding pooled features to match each other using a contrastive loss, as illustrated in Fig. 1. This pooling strategy naturally mitigates two drawbacks encountered in autonomous driving data: (1) It reduces the noise induced by incorrect matching of points and pixels (which is performed automatically), e.g., caused by occlusions for one of the sensors; (2) It balances asymmetries between areas with denser coverage of points and sparser areas, that would otherwise have different weights in the contrastive loss. The latter is particularly important for objects such as cars, pedestrians and cyclists, that are sampled more sparsely than the road near the ego-vehicle.

Finally, we examine key elements of our image-to-Lidar distillation method. This includes a careful design of the image feature projection head to avoid degenerate cases where no useful information is transferred to the 3D network.

In summary, our contributions are the following.

- We propose a novel self-supervised 2D-to-3D representation distillation approach based on a superpixel-to-superpoint contrastive loss and a carefully designed image feature upsampling architecture that allows high resolution image features to be distilled without suffering from degenerate solutions. We call this method SLidR, for Superpixel-driven Lidar Representations.
- To the best of our knowledge, this work provides the first study on the self-supervised image-to-Lidar representation distillation problem for autonomous driving data. This includes extensively evaluating our method for the downstream tasks of semantic segmentation on nuScenes [6] and SemanticKITTI [3] and object detection on KITTI [19], and comparing it against strong baselines. The latter were produced by adapting and optimizing several existing self-supervised pre-training methods for the autonomous driving setting.
- We demonstrate that our image-to-Lidar pre-training strategy surpasses, in all evaluation settings, state-of-the-art 3D self-supervised pre-training methods and prior 2D-to-3D distillation methods, devised for dense point clouds captured in indoor scenes.

2. Related Works

2.1. Self-Supervised Representation Learning

Self-supervised methods aim to learn good representations by pre-training a neural network with an annotation-free pretext task using many unlabeled data. The goal is for these self-supervised representations to transfer well to downstream tasks of interest for which there are limited annotated data available. In the following we review recent self-supervised methods in the image and 3D domain.

2D Self-Supervision. Several approaches have been proposed for representation learning on the image domain [11, 16, 23, 46, 47, 49, 71]. One of the most prominent category of methods are those based on contrastive-based instance discrimination objectives [11, 26, 28, 30, 45, 48, 59, 65], which learn to match different views of the same image data (e.g., generated with random image augmentation) in the presence of distracting, negative examples. Other prominent categories are the feature reconstruction learning methods [9, 13, 21, 22, 24] and clustering-style methods [2, 7, 8, 35, 66, 75]. In this work, we exploit powerful self-supervised image representations, specifically MoCo [12, 28], to pre-train 3D Lidar networks. Apart from image-wise self-supervised objectives, as those already mentioned, there have also been proposed pixel-wise [61, 64, 68, 69] or region-wise self-supervised objectives [31]. Our superpixel-driven image-to-lidar contrastive distillation loss relates to [31, 61] that exploit unsupervised segmentation masks for defining their proposed contrastive objectives.

3D Self-Supervision. Most of the 3D self-supervised methods focus on single objects for tasks such as object recognition or part segmentation. We find techniques based on pretext tasks defined at the object level, based on point cloud reconstruction or prediction of a global transformation [14, 50, 55, 62]. As in 2D self-supervision, some techniques define their pretext task at the feature level using cluster prediction [27], contrastive-based instance discrimination methods [10, 17, 54, 63], a combination of the last two [70], or multimodal object representations [38]. Among methods working on entire scenes rather than on single objects, [33, 41, 67] pre-train a 3D network by learning to match points in two registered point clouds. DepthContrast [72] uses a scene-level instance discrimination pretext task both in indoor and outdoor scenes. Finally, [36, 42] both appeared publicly recently. The first presents a method extensively tested on autonomous driving scenes while the second applies on sequences of point clouds captured indoor or outdoor. Unlike us, none leverages the image modality.

Discussion. For pre-training, self-supervised methods have relied so far on curated and balanced datasets, e.g., ImageNet [53]. Self-supervised methods are highly dependent on the quality of the training data, so the dataset constitutes essentially a form of supervision in itself. In contrast, autonomous driving data, acquired from city streets, is raw and uncurated, displaying strong redundancy and imbalance. Here, self-supervised learning is both challenging and highly necessary to reduce the burden of continuous annotation of data, yet it hasn't been addressed much [60, 72]. We tackle this problem and show that we can overcome some of these challenges by leveraging multi-modality.

2.2. Knowledge Distillation

The purpose of knowledge distillation (KD) is to transfer useful information from a trained teacher network into a student network. To this end, the student is trained to mimic some characteristic of the teacher, e.g., output [32] or intermediate features [52, 59]. Initially used for distilling a large network or ensemble into a smaller network [5, 32, 39], KD has been recently revisited as teacher-student architectures for semi-supervised [40, 57] and unsupervised representation learning [9, 21, 22, 24]. Here, after training the student typically outperforms the teacher.

2D-to-3D knowledge distillation. Our work relates to the setting of KD from a 2D teacher pre-trained on images into a 3D student network [25, 37, 43]. For instance, in [25] indoor RGB-D data is used for distilling an RGB teacher into a 2D student network for depth-maps. The most related to ours is the unpublished concurrent work [43] where 2D representations are distilled in a 3D network. Besides being designed for dense indoor RGB-D data, a major difference between the two methods is that [43] contrasts pixels with points while our method contrasts 2D image regions with 3D point cloud regions, defined using superpixels. We explain the advantages of this superpixel-based distillation formulation in Sec. 3.2. Moreover, in absence of public code, we developed and optimized our best adaptation of this method for autonomous driving data and our empirical comparison with it in Sec. 4 demonstrates the superiority of our method. Finally, the idea of contrasting point-pixel pairs is also exploited in [44] and [34] for 2D-3D modality fusion and building geometry-aware 2D networks, respectively, rather than for KD from a 2D network to a 3D network as done in our method.

3. Our approach

3.1. Image-to-Lidar Self-supervised Distillation

Our goal is to learn, by self-supervised distillation, 3D Lidar representations by leveraging the availability of aligned Lidar and image data in autonomous driving setups.

Synchronized Lidar and image data. Let $\mathsf{P} = (\boldsymbol{p}_i)_{i=1,\dots,n} \in \mathbb{R}^{N\times 3}$ denote a point cloud captured in a scene by a Lidar at time t_0. We assume that C color images $\mathsf{I}_1, \dots, \mathsf{I}_C \in \mathbb{R}^{M\times 3}$, where M denotes the number of pixels,

Figure 2. SLIC superpixels computed on an image from nuScenes

are captured by C cameras in the same scene at t_0 and that the relative poses between the Lidar and cameras sensors are known. The pose information permits us to project each point $\boldsymbol{p}_i$ in the C camera frames. Specifically, we can build a mapping $\rho_c \colon \mathbb{R}^3 \to \{1, \dots, M\} \cup \{0\}$, for each camera c, that takes as input a 3D point $\boldsymbol{p}_i$ and outputs the index of the corresponding 2D pixel in the frame I_c, or 0 if the input point is not viewed in camera c.

Distilling network representations. Let $f_{\theta_{\text{bck}}} \colon \mathbb{R}^{N \times 3} \to \mathbb{R}^{N \times D}$ be a 3D deep neural network, with trainable parameters θ_{bck}, that takes as input a point cloud and outputs one D-dimensional feature per point. Our goal is to pre-train this 3D network without using any human annotations. To this end, we exploit the aligned and synchronized Lidar and image data, described in the previous paragraph. We also leverage the availability of a self-supervised pre-trained image network $g_{\bar{\omega}_{\text{bck}}} \colon \mathbb{R}^{M \times 3} \to \mathbb{R}^{M' \times E}$ with trained and fixed parameters $\bar{\omega}_{\text{bck}}$, that takes as input an image and outputs an E-dimensional feature map at a possibly lower resolution $M' \leqslant M$. In this context, we propose to train $f_{\theta_{\text{bck}}}(\cdot)$ by aligning the point features $f_{\theta_{\text{bck}}}(\mathsf{P})$ with the pre-trained image representations $g_{\bar{\omega}_{\text{bck}}}(\mathsf{I}_1), \dots, g_{\bar{\omega}_{\text{bck}}}(\mathsf{I}_C)$. We achieve this goal with a superpixel-driven contrastive loss, which we describe in the following section.

3.2. Superpixel-driven Contrastive Distillation Loss

To distill the knowledge of the pre-trained image network $g_{\bar{\omega}_{\text{bck}}}(\cdot)$ into the 3D Lidar network $f_{\theta_{\text{bck}}}(\cdot)$ by self-supervision, we use a contrastive loss [48] between the image features obtained from $g_{\bar{\omega}_{\text{bck}}}(\cdot)$ and the 3D point features extracted from $f_{\theta_{\text{bck}}}(\cdot)$. As we are interested in downstream tasks such as semantic segmentation and object detection, the learned 3D representations should "reason" in terms of objects or object parts. We want to contrast features at the object level rather than at the over-detailed pixel level or the overly-coarse scene level.

To that end, we use superpixels for grouping pixels which are locally visually similar, hence likely to belong to one object, and we define our contrastive loss with them. We segment the image I_c into, at most, Q superpixels with SLIC [1], as illustrated in Fig. 2. We denote the superpixels by $\mathcal{S}_1^c, \dots, \mathcal{S}_Q^c$, where $\mathcal{S}_s^c$ is the set of pixel indices belonging to the sth superpixel. We have $\mathcal{S}_1^c \cup \ldots \cup \mathcal{S}_Q^c = \{1, \dots, M\}$ and $\forall s \neq s',\ \mathcal{S}_s^c \cap \mathcal{S}_{s'}^c = \varnothing$. We also use the mapping function ρ_c to group the points viewed in the cth camera into Q distinct superpoints: $\mathcal{G}_1^c, \dots, \mathcal{G}_Q^c$, where $\mathcal{G}_s^c = \{i : \rho_c(\boldsymbol{p}_i) \in \mathcal{S}_s^c\}$. In the ideal case of one object per superpixel, we want the point feature describing $\mathcal{G}_s^c$ to be similar to the image feature describing the corresponding superpixel $\mathcal{S}_s^c$, but unlike the image feature describing a different superpixel $\mathcal{S}_{s'}^{c'}$ from the same scene, for $(s, c) \neq (s', c')$, or the image feature describing a superpixel from a different scene. This distillation principle is formalized as follows.

Contrastive loss. For each camera c, we compute superpoint and superpixel features by average pooling:

$$\boldsymbol{f}_s^c = \frac{1}{|\mathcal{G}_s^c|} \sum_{i \in \mathcal{G}_s^c} (h_{\theta_{\text{head}}} \circ f_{\theta_{\text{bck}}})(\mathsf{P})_i \tag{1}$$

$$\boldsymbol{g}_s^c = \frac{1}{|\mathcal{S}_s^c|} \sum_{j \in \mathcal{S}_s^c} (h_{\omega_{\text{head}}} \circ g_{\bar{\omega}_{\text{bck}}})(\mathsf{I}_c)_j, \tag{2}$$

for all s such that $|\mathcal{G}_s^c| > 0$ (a superpixel feature is computed only if the corresponding superpoint is non-empty). The heads $h_{\theta_{\text{head}}}, h_{\omega_{\text{head}}}$, with trainable parameters θ_{head} and ω_{head}, project respectively the 3D-based and 2D-based features into the same F-dimensional space. The point projection head $h_{\theta_{\text{head}}} \colon \mathbb{R}^{N \times D} \to \mathbb{R}^{N \times F}$ is a simple pointwise linear layer followed by ℓ_2-normalization. The pixel projection head $h_{\omega_{\text{head}}} \colon \mathbb{R}^{M' \times E} \to \mathbb{R}^{M \times F}$ is described in Sec. 3.3.

We transfer the knowledge from the 2D network to the 3D network by using a contrastive loss which favors a solution where a superpoint feature $\boldsymbol{f}_s^c$ is more correlated to its corresponding superpixel features $\boldsymbol{g}_s^c$ than any other feature $\boldsymbol{g}_{s'}^{c'}$, with $(s, c) \neq (s', c')$. Concretely, the network $f_{\theta_{\text{bck}}}$ and the projection heads $h_{\theta_{\text{head}}}, h_{\omega_{\text{head}}}$ are jointly trained using the following superpixel-driven contrastive loss:

$$\mathcal{L}(\theta_{\text{bck}}, \theta_{\text{head}}, \omega_{\text{head}}) = -\sum_{c,s} \log \left[\frac{\exp(\langle \boldsymbol{f}_s^c, \boldsymbol{g}_s^c \rangle / \tau)}{\sum_{c',s'} \exp(\langle \boldsymbol{f}_s^c, \boldsymbol{g}_{s'}^{c'} \rangle / \tau)} \right], \tag{3}$$

where $\langle \cdot, \cdot \rangle$ denotes the scalar product in $\mathbb{R}^F$, and $\tau > 0$ a temperature. For simplicity, we have only defined here the contrastive loss for a single scene. In practice, multiple scenes are available in a batch at training time and the denominator in Eq. (3) actually contains the superpixel features of *all* scenes in the batch.

Discussion. We argue that using a contrastive loss at the superpoint-superpixel level is better adapted for pre-training

$f_{\theta_{\text{bck}}}$ for semantic segmentation and object detection than staying at the point-pixel level.

First, as already mentioned, superpixels permit us to group points and pixels in visually similar regions, thus likely to belong to one object. Hence, the loss (3) will favor point features with the desirable property of being locally coherent when they belong to the same object (assuming the superpixel does not cover several objects). Furthermore, compared to contrasting pixels to points, we lower the proportion of "false negatives" in the contrastive loss as we do not contrast between almost identical points inside a superpixel. Yet, a small number of those false negatives can remain as the superpixels tend to oversegment the objects, but this is common in unsupervised contrastive learning where similar or same class images can be contrasted due to the instance discrimination setting. In addition, our strategy of averaging features within superpixels limits the impact of pixels belonging to different semantic regions. On the other end of the spectrum, scene-level contrastive learning, i.e., contrasting the global representation of an entire point cloud to the global representation(s) of the corresponding camera frames, is not meaningful for autonomous driving data: (a) autonomous driving scenes consist of multiple different objects; (b) there is relatively limited diversity at the scene-level since all scenes consist of almost the same type of objects, e.g., roads, cars, and pedestrians.

Second, superpixels permit us to naturally give the same weights to all regions in the contrastive loss irrespective of the point sampling density in those regions. In typical Lidar scans from AD scenes, the density of points varies greatly with a majority of points sampled on the road and near the ego-vehicle. In the distillation loss of Eq. (3) if we consider points and pixels instead of superpoints and superpixels, we cannot in practice exhaustively consider all possible pairs of matching point and pixel because of computational tractability, and we must subsample them, as done in [43, 67]. However, if this subsampling is performed randomly, without any proper selection, the loss in Eq. (3) is dominated by points in high density regions. With superpixel-based pooling of point features, we reduce the number of pairs, thus removing the need for subsampling, and the loss now treats different objects equally, whether they are seen in a region sampled with high or low density.

Third, the Lidar and image sensors have different viewpoints and the respective acquisitions are never perfectly synchronized. Hence, the point-pixel matching is only approximate, with incorrect matches due to sensor occlusions and motion. Averaging the features as in Eq. (1) and Eq. (2) allows us to reduce the impact of spurious matches.

3.3. The Devil is in the Image Projection Head

In our method, the image network $g_{\bar{\omega}_{\text{bck}}}(\cdot)$ is a ResNet-50 pre-trained under self-supervision on ImageNet [53] using MoCov2 [12, 28]. This network outputs features at a much lower resolution than the resolution of input images: $M' = M/32^2$. To recover features at the pixel level and be able to compute Eq. (2), we tested several architectures for the pixel projection head $h_{\omega_{\text{head}}}$ and were able to reach good performance only when using a 1×1 convolution layer followed by a fixed upsampling method, such as bilinear or nearest neighbor upsampling. Indeed, we noticed that having a $h_{\omega_{\text{head}}}$ architecture that captures wide spatial context for an output pixel feature (e.g., using several convolutional layers with kernel size greater than 1) allows $h_{\omega_{\text{head}}}$ to "cheat" on the contrastive task with the "leaked" context information, by matching a 2D image feature with its paired 3D point(s) only based on the 2D spatial position of the 2D feature inside the image. This would allow the network to solve the contrastive task, however it is of no use for learning high-level semantic features. We expect similar degenerate solutions if $g_{\bar{\omega}_{\text{bck}}}(\cdot)$ is not kept frozen.

Empirically, we noticed that performance improves as we preserve the input resolution in the ResNet-50 encoder. We propose a few simple adjustments to the ResNet-50 encoder to limit downsampling without increasing significantly the computational complexity or adding new parameters. We keep the first strided convolution and max pooling layer, but then maintain a fixed resolution across all residual blocks by replacing strided convolutions with dilated convolutions. This leads to the lesser discrepancy $M' = M/4^2$.

To conclude, to bridge the gap between the two resolutions, the pixel projection head $h_{\omega_{\text{head}}}$ consists of a pixel-wise convolution, followed by a bilinear upsampling layer by 4 in each spatial direction, and an ℓ_2-normalization layer.

4. Experiments

4.1. 3D Network Pre-training

Network backbones. The 3D backbone $f_{\theta_{\text{bck}}}$ is the sparse residual U-Net architecture used in [67], which was originally designed in [15], where we use $3 \times 3 \times 3$ kernels for all sparse convolutions. As input representation it takes a sparse occupancy grid of the 3D data obtained by quantizing the 3D points, i.e., voxels. Instead of using voxel in Cartesian coordinates as in [43, 67, 72], we use voxels in cylindrical coordinates which are better suited for Lidar point clouds [74]. The voxel size that we use is 10 cm on the z-axis and radius (distance to the origin in the xy-plane) component, and 1° on the azimuth angle. The 2D backbone is a ResNet-50 [29] pre-trained with MoCov2 [12, 28].

Pre-training dataset. We pre-train all models on nuScenes [6]. This dataset contains 700 training scenes from which we keep aside 100 scenes that constitute our mini-val split, used to choose our training hyperparameters and do our ablation study. The models are pre-trained using all the keyframes from the 600 remaining training scenes.

Method	Dil. Conv.	Superpix.	mIoU
PPKT† [43]	✗	✗	34.7
SLidR w/o superpix.	✓	✗	36.6 (+1.9)
SLidR	✓	✓	**39.2** **(+4.5)**

Table 1. Ablation study on nuScenes semantic segmentation by replacing (3) with a pixel-level contrastive loss (SLidR w/o superpix.) and then using strided convolution instead of dilated convolution in ResNet-50 (PPKT†). The scores are obtained by linear probing of the pre-trained backbone. We report the mIoU on our mini-val split.

Initialization of $f_{\theta_{\text{bck}}}$	nuScenes [6]		KITTI [3]
	Lin. prob. 100%	Finetuning 1%	Finetuning 1%
Random	8.1	30.3	39.5
PointContrast† [67]	21.9	32.5 (+2.2)	41.1 (+1.6)
DepthContrast† [72]	22.1	31.7 (+1.9)	41.5 (+1.2)
PPKT† [43]	36.4	37.8 (+7.5)	43.9 (+4.4)
SLidR	**38.8**	**38.3** **(+8.0)**	**44.6** **(+5.1)**

Table 2. Comparison of different pre-training methods for semantic segmentation by linear probing or finetuning. On nuScenes [6], we use either 1% or 100% of the annotated training scans. On SemanticKITTI [3], we use 1% of the annotated training scans. We report the mIoU on the validation set of nuScenes and on the sequence 8 of SemanticKITTI.

Training parameters. Unless mentioned otherwise, $f_{\theta_{\text{bck}}}$, $h_{\theta_{\text{head}}}$ and $h_{\omega_{\text{head}}}$ are trained on 1 GPU with SLidR for 50 epochs using SGD with a initial learning rate of 0.5, a momentum of 0.9, weight decay of 0.0001, dampening of 0.1, and a cosine annealing scheduler that decreases the learning rate from its initial value to 0 at the end of the 50th epoch.

Data augmentation. A key factor in the success of self-supervision is the use of strong data augmentations [12, 22, 72]. We apply several augmentations detailed in the supplementary material. In summary, on the point cloud side, we apply a random rotation around the z-axis, flip randomly the direction of the x and y-axis, and drop points that lie in a random cuboid as in [72]. On the image side, we use a random horizontal flip and a random crop-resize.

4.2. Baselines

To the best of our knowledge, this is the first work to study image-to-Lidar self-supervised representation distillation on autonomous driving data. Hence, we cannot rely on existing baselines trained in this setup. In order to fairly compare against strong baselines, a significant amount of work was done to adapt and optimize existing pre-training methods to our setup. In particular, we use 3 representative methods for 3D network pre-training as baselines: PPKT [43], PointContrast [67], and DepthContrast [72].

PPKT for autonomous driving. The first baseline is PPKT, which is also a 2D-to-3D representation distillation method, but based on a pixel-to-point contrastive loss. Since PPKT was originally proposed for RGB-D data captured indoor and, up to now, there is no publicly released code, we propose our best adaption of this method in an autonomous driving setup, referred as PPKT†.

PointContrast. The second baseline is PointContrast [67], which, rather than image-to-lidar distillation, learns 3D representations with a contrastive task defined only at the level of 3D points. We retrained PointContrast on nuScenes after studying several setups to optimize its performance. This method requires pairs of point clouds acquired from different viewpoints in the same scene with a list of matching points in these two views. We use pairs of nuScenes' keyframes in each scene and provide the details of the construction of this dataset in the supplementary material. We denote this adapted version PointContrast†.

DepthContrast. The last baseline is DepthContrast [72] which pre-trains simultaneously two 3D network backbones, a point-based network, e.g., [51] and a voxel-based network, e.g., [73], using a contrastive task between the global point-cloud representations of the two networks. To make it comparable with the rest of the methods, we used the point-based network of [72] and used our sparse residual U-Net that processes occupancy map in cylindrical coordinate as the voxel-based network. We only evaluate the performance of this voxel-based network after pre-training.

Training setup. All the baselines are pre-trained on the same nuScenes split as for SLidR, using 1 GPU, and after tuning the hyperparameters for our min-val split, except for DepthContrast for which the performance are significantly improved when using 4 GPUs. We use the same data augmentations, voxel-based 3D network backbone, and cylindrical coordinate voxels as in our method. More implementation details are provided in the supplementary material.

4.3. Transfer on Semantic Segmentation

We study the quality of the learned representations obtained with SLidR and the baselines for semantic segmentation. We test the performance of these methods on nuScenes [6], the dataset viewed during pre-training, but also test the robustness to a domain change by using SemanticKITTI [3]. There are 16 semantic classes in nuScenes and 19 in SemanticKITTI. Except otherwise mentioned, the quality of the models is evaluated on the original validation split of nuScenes and on the sequence 8 of SemanticKITTI.

Evaluation settings. We use two evaluation protocols. In

Init. of $f_{\theta_{\text{bck}}}$	1%	5%	10%	25%	100%
Random	30.3	47.7	56.6	64.8	74.2
SLidR	**39.0** (+8.7)	**52.2** (+4.5)	**58.8** (+2.2)	**66.2** (+1.4)	**74.6** (+0.4)

Table 3. Improvement of the performance thanks to SLidR for semantic segmentation over a randomly initialized network as a function of the percentage of available annotations on nuScenes. We report the mIoU on the validation set of nuScenes.

Init. of $f_{\theta_{\text{bck}}}$	5%	10%	20%
Random	56.1	59.1	61.6
PPKT†	**57.8** (+1.7)	60.1 (+1.0)	61.2 (-0.4)
SLidR	**57.8** (+1.7)	**61.4** (+2.3)	**62.4** (+0.8)

Table 4. Performance of pre-training methods for object detection by fine tuning pre-trained networks using different percentages of the annotated scans in the KITTI 3D object detection dataset [19]. Scores are average mAP across cars, pedestrians and cyclists.

both, we adapt the pre-trained 3D backbones $f_{\theta_{\text{bck}}}$ for the semantic segmentation task by adding a point-wise linear classification head on their output. The first protocol evaluates the quality of the pre-trained features as they are, by linear probing them. To that end, we only train the added classification head on the nuScenes [6] dataset, while keeping the pre-trained parameters of $f_{\theta_{\text{bck}}}$ fixed. The second protocol evaluates the ability of the pre-trained 3D representations to learn to perform semantic segmentation in a regime where only a small number of annotations is available. Therefore, in this protocol, we fine-tune the entire network on the semantic segmentation task on nuScenes or semanticKITTI [3] using only a portion of the available annotations. In both protocols, we use a linear combination of the cross-entropy and the Lovász loss [4] as training objective. For few-shot semantic segmentation, we optimized the fine-tuning learning rate for each method using our mini-val split for nuScenes and 10 percent of the training set of semanticKITTI, which remained unused during few-shot fine-tuning. Training details are provided in the supplementary material.

4.3.1 Ablation Study

We first justify the use of dilated convolutions in the ResNet-50 and of our superpixel-driven contrastive loss (3). To that end, we train 3 networks with: (a) SLidR, (b) SLidR without superpixels (i.e., using a point-to-pixel version of (3)), and finally (c) SLidR without superpixels and without the dilated convolution, which is our implementation of PPKT†. We then test the quality of the trained backbone by linear probing for semantic segmentation on nuScenes.

The results computed on our mini-val split are reported in Tab. 1 and show that each of our technical contributions improve the performance. We notice a gain of 1.9 point thanks to the dilated convolutions and, most importantly, a huge improvement of 4.5 point when combined with our key ingredient: the superpixel-driven contrastive loss (3).

A complementary analysis of the sensitivity of SLidR to the choice of the underlying superpixel algorithm is available in the supplementary material.

4.3.2 Comparisons with Baselines

We compare SLidR with the baselines on the linear probing setup on nuScenes and the few-shot end-to-end fine-tuning setup using 1% of the available annotations on nuScenes and SemanticKITTI. The results are reported in Tab. 2.

We observe that: **(1)** All pre-training methods are better than random initialization. **(2)** PPKT† and our SLidR, the methods based on image-to-Lidar distillation for pre-training, perform significantly better than the DepthContrast† and PointContrast† approaches. This highlights the advantage of exploiting self-supervised image pre-trained networks for learning 3D Lidar representations, as we advocate in our work. **(3)** SLidR achieves better semantic segmentation performance, especially on the linear probing setup, which demonstrates the superiority of our superpixel-driven method.

4.3.3 Annotation Efficiency on nuScenes

We continue by studying the performance of SLidR compared to a training from a random initialization as function of the percentage of annotated data for semantic segmentation on nuScenes. We fine-tune end-to-end the two examined networks using 1%, 5%, 10%, 25% or 100% of the annotations on nuScenes, with learning rates optimized on our mini-val split for each method and subset size.

The results are presented in Tab. 3 and show a constant improvement over a training from a randomly initialized backbone. The improvement is 0.4 point on the mIoU for 100% annotations and increases to up to 8.7 point as the percentage of available annotations decreases.

4.4. Transfer for Few-Shot Object Detection

Here we evaluate the quality of our pre-trained Lidar representations on the challenging downstream tasks of 3D object detection on the KITTI dataset [19].

Experimental setup. We use OpenPCDet [58] in which we modify the PointRCNN model by replacing the PointNet++ [51] backbone with our pre-trained backbone. This model is fine-tuned on subsets of different sizes of the standard KITTI object detection dataset [19], which contains

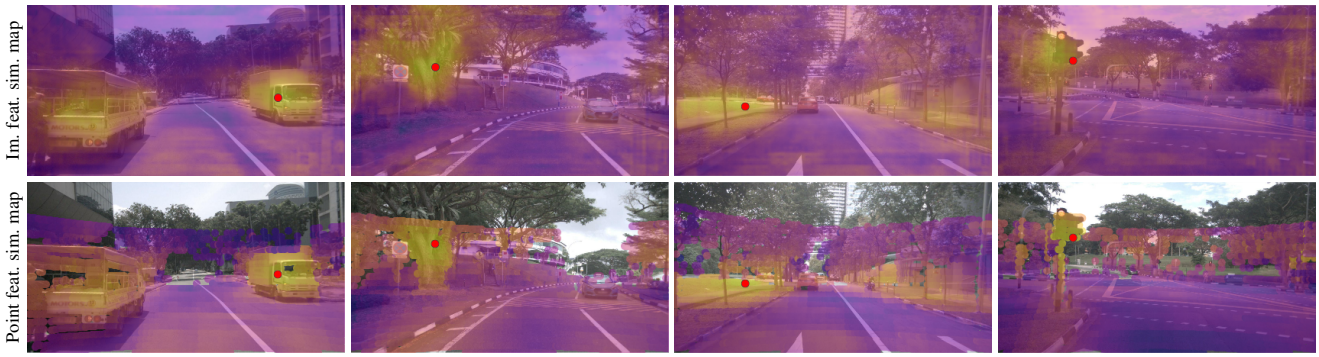

Figure 3. We present the cosine similarity between the SLidR's feature of a query point (displayed as a red dot) and: (a) the pixel features of an image in the same scene (top row - Image feature similarity map); (b) the features of the other points projected in the same image (bottom row - Point feature similarity map). The colormap goes from violet to yellow for respectively low and high similarity scores. We show these maps for two scenes in the validation set of [6].

bounding boxes for cars, cyclists and pedestrians. The performance are compared by computing the average mAP over these three classes in the moderately difficult cases, which are used to rank all methods on this dataset [19].

Fine-tuning protocol. We initialize the pre-trained backbone $f_{\theta_{\text{bck}}}$ with SLidR, PPKT†, or using random weights. For this experiments, $f_{\theta_{\text{bck}}}$ have been pre-trained using 4 GPUs, a batch size of 16, a initial learning rate of 2, and synchronized batch-norm layers for PPKT† and SLidR. The networks are then fine-tuned on 4 GPUs with a batch size of 12 and the default settings of OpenPCDet. The learning rate for fine-tuning is optimized for each subset of the KITTI object detection dataset so as to maximize the performance of the backbone initialized with random weights. We then use the same learning rate for the other methods.

Results. The results are reported in Tab. 4. We remark that SLidR gives a significant improvement of up to 2.3 points in mAP over a situation where no pre-training is done. SLidR also outperforms PPKT† with a gain of at least 1.2 point in mAP at 10% and 20% of annotated data.

4.5. Visual Inspection

The feature similarity maps presented in Fig. 3 highlights our pre-trained model's object recognition and segmentation abilities, by showing how features are locally coherent when they belong to the same object. In the leftmost scene, the query point on the truck on the right side is highly correlated with points on the same truck as well as with points on the truck on the left side. This indicates that our self-supervised 3D features already allows distinction of objects without fine-tuning. The same phenomenon is observed with points on the traffic lights in the rightmost scene. Furthermore, the nearly identical image and point feature similarity maps illustrate the quality of the knowledge transfer. Finally, we remark some spurious correlations on the road, which indicate that SLidR might still be improved.

4.6. Technical Limitations

A first limitation might occur in low-light conditions as the computed superpixels might provide irrelevant object segments, impairing the performance of our method.

Another limitation occurs when the output image features are similar between two superpixels, e.g, $\boldsymbol{g}_1^c \approx \boldsymbol{g}_2^c$, as then, the contrastive loss will try to enforce a solution where the superpoint feature $\boldsymbol{f}_1^c$ is correlated to $\boldsymbol{g}_1^c$ but uncorrelated to $\boldsymbol{g}_2^c$, which is impossible. While this issue is common in contrastive self-supervised methods, the impact is possibly a bit stronger here as the whole image backbone $g(\cdot)$ is frozen, leaving less room for adjustments in these situations. Addressing this limitation is left for future work.

5. Conclusion

We proposed SLidR, a self-supervised image-to-Lidar distillation method working on synchronized Lidar and camera data, as typically found in autonomous driving setups. The key ingredient of our method is the use of superpixels to produce object-aware point representation suited for, e.g., semantic segmentation and object detection. We showed that SLidR yields powerful point cloud representations which transfer and generalize well to multiple tasks and datasets, surpassing related state-of-the-art methods.

Acknowledgments. We thank Antonin Vobecky, David Hurych, Josef Sivic and Patrick Pérez for their feedbacks and the fruitful discussions around this project.

References

[1] R. Achanta, A. Shaji, Kevin Smith, Aurélien Lucchi, P. Fua, and S. Süsstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE TPAMI*, 34:2274–2282, 2012. 2, 4, 15

[2] Yuki M Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. In *ICLR*, 2020. 3

[3] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. SemanticKITTI: A dataset for semantic scene understanding of lidar sequences. In *ICCV*, pages 9297–9307, 2019. 1, 2, 6, 7

[4] Maxim Berman, Amal Rannen Triki, and Matthew B Blaschko. The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In *CVPR*, pages 4413–4421, 2018. 7

[5] Cristian Buciluă, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *SIGKDD*, pages 535–541, 2006. 3

[6] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *CVPR*, pages 11618–11628, 2020. 2, 5, 6, 7, 8, 12, 13, 14

[7] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and M. Douze. Deep clustering for unsupervised learning of visual features. In *ECCV*, pages 132–149, 2018. 3

[8] Mathilde Caron, Ishan Misra, J. Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *NeurIPS*, pages 9912–9924, 2020. 2, 3

[9] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, pages 9650–9660, 2021. 3, 12

[10] Haolan Chen, Shitong Luo, Xiang Gao, and Wei Hu. Unsupervised learning of geometric sampling invariant representations for 3D point clouds. In *ICCV*, pages 893–903, 2021. 3

[11] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, pages 1597–1607, 2020. 3

[12] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv:2003.04297*, 2020. 3, 5, 6, 12

[13] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *CVPR*, pages 15750–15758, 2021. 3

[14] Ye Chen, Jinxian Liu, Bingbing Ni, Hang Wang, Jiancheng Yang, Ning Liu, Teng Li, and Qi Tian. Shape self-correction for unsupervised point cloud understanding. In *ICCV*, pages 8382–8391, 2021. 3

[15] Christopher B. Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *CVPR*, pages 3075–3084, 2019. 5

[16] Carl Doersch, Abhinav Gupta, and Alexei Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, pages 1422–1430, 2015. 3

[17] Bi'an Du, Xiang Gao, Wei Hu, and Xin Li. Self-contrastive learning with hard negative sampling for self-supervised point cloud learning. In *ACM MM*, pages 3133–3142, 2021. 3

[18] Pedro F. Felzenszwalb and D. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 59:167–181, 2004. 2, 12, 15

[19] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *CVPR*, pages 3354–3361, 2012. 2, 7, 8

[20] Kyle Genova, Xiaoqi Yin, Abhijit Kundu, Caroline Pantofaru, Forrester Cole, Avneesh Sud, Brian Brewington, Brian Shucker, and Thomas Funkhouser. Learning 3D semantic segmentation with only 2D image supervision. In *3DV*, pages 361–372, 2021. 1

[21] Spyros Gidaris, Andrei Bursuc, Nikos Komodakis, Patrick Pérez, and Matthieu Cord. Learning representations by predicting bags of visual words. In *CVPR*, pages 6926–6936, 2020. 3

[22] Spyros Gidaris, Andrei Bursuc, Gilles Puy, Nikos Komodakis, Matthieu Cord, and Patrick Pérez. OBoW: Online bag-of-visual-words generation for self-supervised learning. In *CVPR*, pages 6826–6836, 2021. 2, 3, 6

[23] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018. 3

[24] Jean-Bastien Grill, Florian Strub, Florent Altch'e, C. Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, B. A. Pires, Z. Guo, M. G. Azar, Bilal Piot, K. Kavukcuoglu, R. Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. In *NeurIPS*, pages 21271–21284, 2020. 2, 3

[25] Saurabh Gupta, Judy Hoffman, and Jitendra Malik. Cross modal distillation for supervision transfer. In *CVPR*, pages 2827–2836, 2016. 3

[26] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR*, page 1735–1742, 2006. 3

[27] Kaveh Hassani and Mike Haley. Unsupervised multi-task feature learning on point clouds. In *ICCV*, pages 8159–8170, 2019. 3

[28] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, pages 9726–9735, 2020. 2, 3, 5

[29] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 5

[30] Olivier Henaff. Data-efficient image recognition with contrastive predictive coding. In *ICML*, pages 4182–4192, 2020. 3

[31] Olivier J. Hénaff, Skanda Koppula, Jean-Baptiste Alayrac, Aaron van den Oord, Oriol Vinyals, and João Carreira. Efficient visual pretraining with contrastive detection. In *ICCV*, pages 10086–10096, 2021. 2, 3

[32] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NeurIPSW*, 2014. 3

[33] Ji Hou, Benjamin Graham, Matthias Niessner, and Saining Xie. Exploring data-efficient 3d scene understanding with contrastive scene contexts. In *CVPR*, pages 15587–15597, 2021. 3

[34] Ji Hou, Saining Xie, Benjamin Graham, Angela Dai, and Matthias Nießner. Pri3D: Can 3D priors help 2D representation learning? In *ICCV*, pages 5693–5702, 2021. 3

[35] Jiabo Huang, Qi Dong, and Shaogang Gong. Unsupervised deep learning by neighbourhood discovery. In *ICML*, pages 2849–2858, 2019. 3

[36] Siyuan Huang, Yichen Xie, Song-Chun Zhu, and Yixin Zhu. Spatio-temporal self-supervised representation learning for 3D point clouds. In *ICCV*, pages 6535–6545, 2021. 3

[37] Maximilian Jaritz, Tuan-Hung Vu, Raoul de Charette, Emilie Wirbel, and Patrick Pérez. xMUDA: Cross-modal unsupervised domain adaptation for 3D semantic segmentation. In *CVPR*, pages 12605–12614, 2020. 3

[38] Longlong Jing, Ling Zhang, and Yingli Tian. Self-supervised feature learning by cross-modality and cross-view correspondences. In *CVPR*, pages 1581–1591, 2021. 3

[39] Anoop Korattikara Balan, Vivek Rathod, Kevin P Murphy, and Max Welling. Bayesian dark knowledge. In *NeurIPS*, pages 3438–3446, 2015. 3

[40] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. In *ICLR*, 2017. 3

[41] Shamit Lal, Mihir Prabhudesai, Ishita Mediratta, Adam W. Harley, and Katerina Fragkiadaki. CoCoNets: Continuous contrastive 3D scene representations. In *CVPR*, pages 12487–12496, 2021. 3

[42] Hanxue Liang, Chenhan Jiang, Dapeng Feng, Xin Chen, Hang Xu, Xiaodan Liang, Wei Zhang, Zhenguo Li, and Luc Van Gool. Exploring geometry-aware contrast and clustering harmonization for self-supervised 3D object detection. In *ICCV*, pages 3293–3302, 2021. 3

[43] Yueh-Cheng Liu, Yu-Kai Huang, HungYueh Chiang, Hung-Ting Su, Zhe Yu Liu, Chin-Tang Chen, Ching-Yu Tseng, and Winston H. Hsu. Learning from 2D: Pixel-to-point knowledge transfer for 3D pretraining. *arxiv:2104.04687*, 2021. 3, 5, 6, 14

[44] Yunze Liu, Li Yi, Shanghang Zhang, Qingnan Fan, Thomas A. Funkhouser, and Hao Dong. P4Contrast: Contrastive learning with pairs of point-pixel pairs for RGB-D scene understanding. *arxiv:2012.13089*, 2020. 3

[45] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *CVPR*, pages 6707–6717, 2020. 3

[46] Ishan Misra, Lawrence Zitnick, and Martial Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *ECCV*, pages 527–544, 2016. 3

[47] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, pages 69–84, 2016. 3

[48] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arxiv:1807.03748*, 2018. 3, 4

[49] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei Efros. Context encoders: Feature learning by inpainting. In *CVPR*, pages 2536–2544, 2016. 3

[50] Omid Poursaeed, Tianxing Jiang, Quintessa Qiao, Nayun Xu, and Vladimir G. Kim. Self-supervised learning of point clouds via orientation estimation. In *3DV*, pages 1018–1028, 2020. 3

[51] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, pages 5099–5108, 2017. 6, 7, 15

[52] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. FitNets: Hints for thin deep nets. In *ICLR*, 2015. 3

[53] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015. 3, 5

[54] Aditya Sanghi. Info3d: Representation learning on 3D objects using mutual information maximization and contrastive learning. In *ECCV*, pages 626–642, 2020. 3

[55] Jonathan Sauder and Bjarne Sievers. Self-supervised deep learning on point clouds by reconstructing space. In *NeurIPS*, pages 12962–12972, 2019. 3

[56] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *CVPR*, pages 2443–2451, 2020. 15

[57] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NeurIPS*, pages 1195–1204, 2017. 3

[58] OpenPCDet Development Team. OpenPCDet: An open-source toolbox for 3D object detection from point clouds. https://github.com/open-mmlab/OpenPCDet, 2020. 7

[59] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. In *ICLR*, 2020. 3

[60] Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, and Luc Van Gool. Revisiting contrastive methods for unsupervised learning of visual representations. In *NeurIPS*, 2021. 3

[61] Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, and Luc Van Gool. Unsupervised semantic segmentation by contrasting object mask proposals. In *ICCV*, pages 10052–10062, 2021. 3

[62] Hanchen Wang, Qi Liu, Xiangyu Yue, Joan Lasenby, and Matt J Kusner. Unsupervised point cloud pre-training via occlusion completion. In *ICCV*, pages 9782–9792, 2021. 3

[63] Peng-Shuai Wang, Yu-Qi Yang, Qian-Fang Zou, Zhirong Wu, Yang Liu, and Xin Tong. Unsupervised 3D learning for shape analysis via multiresolution instance discrimination. In *AAAI*, pages 2773–2781, 2021. 3

[64] Xinlong Wang, Rufeng Zhang, Chunhua Shen, Tao Kong, and Lei Li. Dense contrastive learning for self-supervised visual pre-training. In *CVPR*, pages 3024–3033, 2021. 3

[65] Zhirong Wu, Yuanjun Xiong, Stella Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance-level discrimination. In *CVPR*, pages 3733–3742, 2018. 3

[66] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *ICML*, pages 478–487, 2016. 3

[67] Saining Xie, Jiatao Gu, Demi Guo, Charles R. Qi, Leonidas J. Guibas, and Or Litany. PointContrast: Unsupervised pre-training for 3D point cloud understanding. In *ECCV*, pages 574–591, 2020. 1, 3, 5, 6, 15

[68] Zhenda Xie, Yutong Lin, Zheng Zhang, Yue Cao, Stephen Lin, and Han Hu. Propagate yourself: Exploring pixel-level consistency for unsupervised visual representation learning. In *CVPR*, pages 16684–16693, 2021. 3

[69] Yuwen Xiong, Mengye Ren, Wenyuan Zeng, and Raquel Urtasun. Self-supervised representation learning from flow equivariance. In *ICCV*, pages 10191–10200, 2021. 3

[70] Ling Zhang and Zhigang Zhu. Unsupervised feature learning for point cloud understanding by contrasting and clustering using graph convolutional neural networks. In *3DV*, pages 395–404, 2019. 3

[71] Richard Zhang, Phillip Isola, and Alexei Efros. Colorful image colorization. In *ECCV*, pages 649–666, 2016. 3

[72] Zaiwei Zhang, Rohit Girdhar, Armand Joulin, and Ishan Misra. Self-supervised pretraining of 3D features on any point-cloud. In *ICCV*, pages 10252–10263, 2021. 1, 3, 5, 6, 13, 15

[73] Yin Zhou and Oncel Tuzel. VoxelNet: End-to-end learning for point cloud based 3D object detection. In *CVPR*, pages 4490–4499, 2018. 6, 15

[74] 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 *CVPR*, pages 9934–9943, 2021. 5

[75] Chengxu Zhuang, Alex Lin Zhai, and Daniel Yamins. Local aggregation for unsupervised learning of visual embeddings. In *CVPR*, pages 6001–6011, 2019. 3

Supplementary Material

A. Complementary Results 12
- A.1. Visual Inspection 12
- A.2. Choice of the Image Backbone 12
- A.3. Choice of the Superpixels Method 12
- A.4. Few-Shot Semantic Segmentation 13

B. Data Augmentations 13

C. Baselines' Implementations Details 14
- C.1. PPKT for Autonomous Driving. 14
- C.2. PointContrast 15
- C.3. DepthContrast 15

D. Additional Training Details 15
- D.1. Linear Probing 15
- D.2. Few-Shot Semantic Segmentation 15
- D.3. Annotation Efficiency on nuScenes 16

E. nuScenes mini-val Split 16

F. Societal and Environmental Impact 16

G. Public Resources Used 16

A. Complementary Results

A.1. Visual Inspection

We present in Fig. 4 and Fig. 5 additional feature similarity maps such as those presented in Fig. 3 in the main paper. We continue to observe the segmentation ability of our pre-trained model: a query point on a tree, road, vehicle, bike is mostly correlated with other points or pixels on trees, road, vehicles, bikes, respectively.

We also notice some spurious correlations with points or pixels around the objects. These spurious correlations seem more apparent in the image feature similarity maps. This can be due to the reduced resolution of the image feature maps (1/4 in each spatial direction) or that the ResNet-50 features are intrinsically imprecise near object boundaries, which prevent the network to learn accurate near object edges in the 3D point cloud. Increasing the image resolution as well as adding additional constraints leveraging the 3D structures observed in the point cloud can help us to prevent such leakage in future works.

We also provide in the supplementary material a video illustrating the capacity of our pre-trained model in doing semantic segmentation on a sequence of point clouds, without fine-tuning. The video is generated as follows. We choose a scene from the validation set of nuScenes [6]. We collect all the point features along with the class labels in the first frame of the sequence. This set constitutes our annotated database. We consider three classes: 'vegetation', 'car' and 'pedestrian'. The points in the following frames are classified by using a binary k-NN classifier (k=20) for each of these class. We display the predicted probability for each class in all subsequent frames: red for 'car', blue for 'pedestrian', green for 'vegetation'.[1] We notice that we are able to classify correctly several points in of each of the considered classes throughout the whole sequence. In particular, we detect correctly pedestrians at the beginning of sequence and cars at the end of the sequence. Some points are misclassified but we recall these results are obtained without any fine-tuning of the backbone.

Backbone $g_{\bar{\omega}_{\text{bck}}}$	Pretrain	mIoU
ResNet-50	Full sup.	39.2
ResNet-50	MoCov2 [12]	39.2
ViT-S/16	DINO [9]	39.5

Table 5. Performance of SLidR on nuScenes semantic segmentation with using different image backbone architectures or pre-training methods. We consider two architectures, ResNet-50 or ViT-S/16, and pretraining under full supervision on ImageNet or by self-supervision (MoCov2, DINO). The scores are obtained by linear probing of the pre-trained backbone. We report the mIoU on our mini-val split.

A.2. Choice of the Image Backbone

We present in Tab. 5 the performance reached with SLidR on nuScenes semantic segmentation when using a ResNet-50 pre-trained under full supervision on ImageNet or under self-supervision using MoCov2. We notice that there is no loss of performance because of the use of self-supervision to pretrain $g_{\bar{\omega}_{\text{bck}}}$.

We also report in the same table the performance obtained when using a self-supervised transformer [9] as image backbone. The performance is slightly higher than when using a ResNet-50, showing that our method is compatible with this type of image network architectures. SLidR can exploit the higher capacity of transformers in learning image representations, which can in turn yields better 3D networks.

A.3. Choice of the Superpixels Method

We present in Tab. 6 the impact of the choice of the number of superpixels or of the superpixels algorithm on the performance of SLidR. For Felzenszwalb's [18] method (called FH), we used a scale parameter of 300, a gaussian pre-processing of standard deviation 0.35 and a superpixels minimal size of 4000 pixels, which yield at most 143 superpixels per image on nuScenes' training set. For SLIC, we tested 100, 150 or 200 superpixels per image. We see that it is important to adjust the number of superpixels correctly to avoid too much over-segmentation and under-segmentation which both impact negatively the performance. The results

[1] A mix of these colors corresponds to predictions with non-zero probability in two or more of these classes.

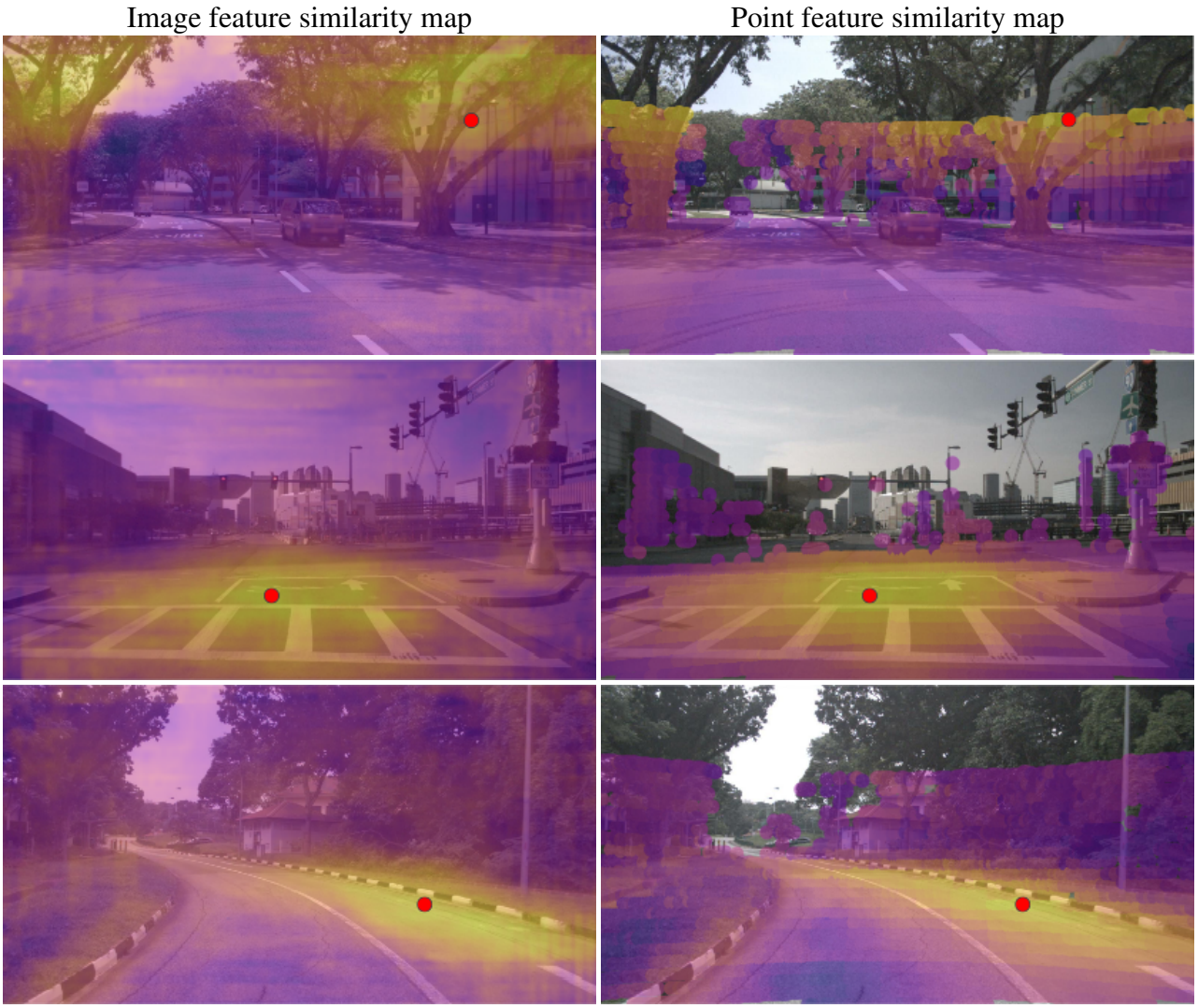

Figure 4. Cosine similarity between the SLidR's feature of a query point (displayed as a red dot) and: (a) the pixel features of an image in the same scene (left column - image feature similarity map); (b) the features of the other points projected in the same image (right column - point feature similarity map). The colormap goes from violet to yellow for respectively low and high similarity scores. We show these maps for two scenes in the validation set of [6].

in the main paper are obtained with SLIC and 150 superpixels per image. Finally, we notice that SLidR is less sensitive to the choice of superpixel algorithm (e.g., using FH instead of SLIC) once its parameters are set correctly.

A.4. Few-Shot Semantic Segmentation

We report in Tab. 7 and Tab. 8 the per-class performance of SLidR and the different baselines when pretraining on 1% of the available annotations. We notice that PPKT† and SLidR are the two best methods on the majority of the classes with SLidR achieving the highest mIoU. On nuScenes, SLidR is ranked first on 9 classes vs 6 for PPKT†. On SemanticKITTI, SLidR is ranked first on 11 of the classes vs 5 for PPKT†.

B. Data Augmentations

As mentioned in Sec. 4, we apply two sets of strong data augmentations: the first on point clouds, the second on images. We highlight that implementing these augmentations is not trivial as they impact the list of point-pixel correspondences, which needs to be updated appropriately.

Regarding point clouds, we apply a random rotation around the z-axis and flip the direction of the x and y-axis with 50% probability for each axis. As in [72], we also drop points that lie in an axis-aligned random cuboid. Concretely, the cuboid center is placed on a randomly-selected

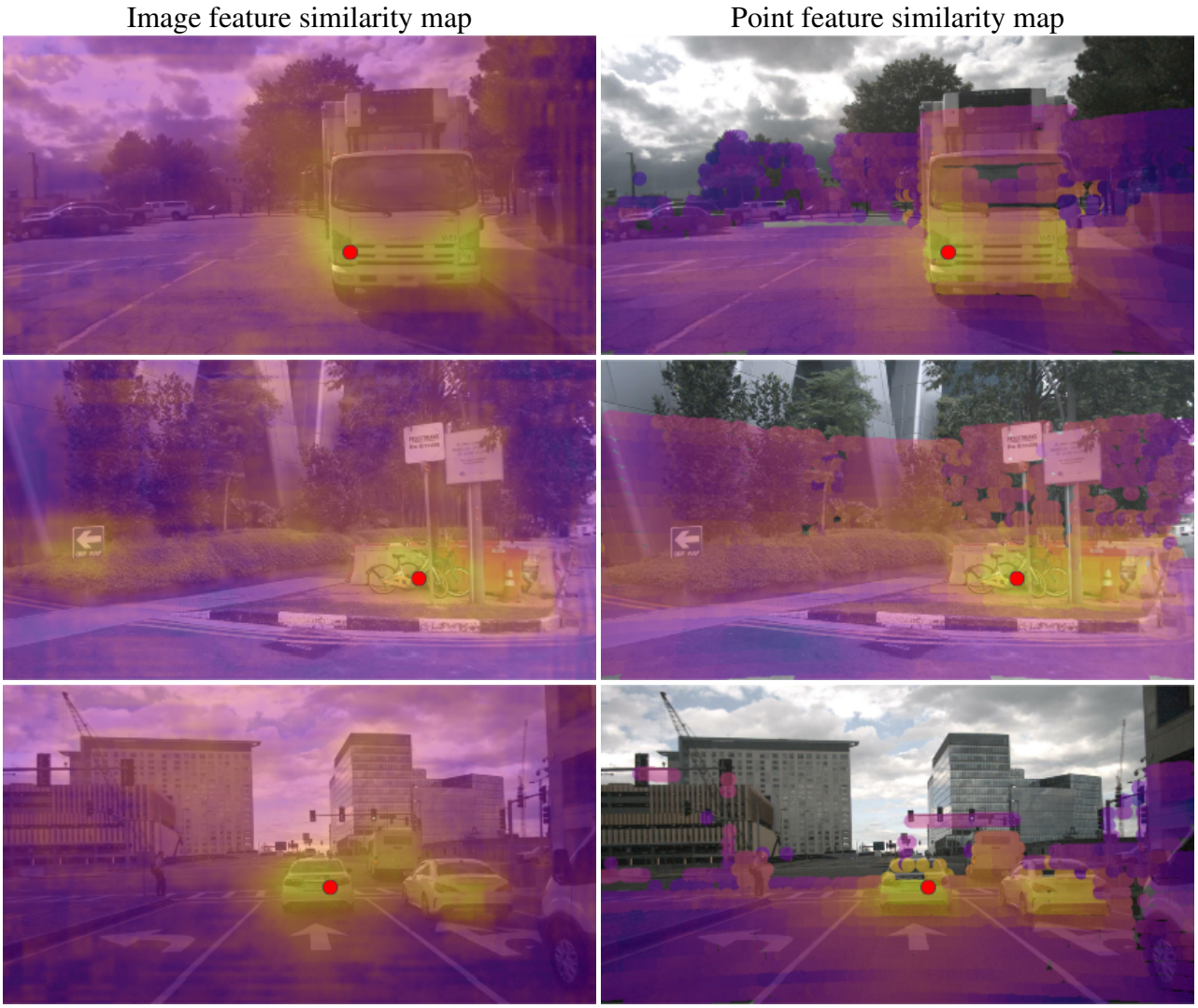

Figure 5. Cosine similarity between the SLidR's feature of a query point (displayed as a red dot) and: (a) the pixel features of an image in the same scene (left column - image feature similarity map); (b) the features of the other points projected in the same image (right column - point feature similarity map). The colormap goes from violet to yellow for respectively low and high similarity scores. We show these maps for two scenes in the validation set of [6].

point in the point cloud P and the length of each side covers at most 10% of the range of point coordinates on the corresponding axis. We make sure that this dropped cuboid preserves at least 1024 pairs of points and pixels, otherwise another new cuboid is selected.

The images are flipped horizontally 50% of the time, and cropped-resized to 416×224. The random crop covers at least 30% of the image area with a random aspect ratio between $14/9$ and $17/9$ before resizing. We make sure that this random cropping preserves at least 1024 or 75% of the pixel-point pairs, otherwise another crop is selected.

C. Baselines' Implementations Details

C.1. PPKT for Autonomous Driving.

PPKT [43] was originally proposed for RGB-D data captured indoor. As, up to now, there is no publicly released code, we propose our best adaption of this method in an autonomous driving setup, referred as PPKT†. PPKT† uses the same data augmentations, voxel-based 3D network backbone, and cylindrical coordinate voxels as our method.

PPKT† is obtained by: using $Q = M$ in Sec. 3, i.e., each superpixel contains one pixel; using strided convolutions in the image backbone $g_{\bar{\omega}_{\text{bck}}}(\cdot)$; the same image head $h_{\omega_{\text{head}}}$ as ours but with a bilinear upsamling layer from a res-

Algorithm	None	SLIC [1]			FH [18]
#Superpixels		100	150	200	$\leqslant$143
mIoU	36.6	37.7	39.2	36.3	39.2

Table 6. Sensitivity of SLidR to the superpixel algorithms and superpixel parameters. The semantic segmentation scores (mIoU) are obtained by linear probing and computed on our nuScenes mini-val split. We compare the performance of SLidR when using (a) no superpixels; (b) SLIC [1] with different number of superpixels per image; and (c) the Felzenszwalb's [18] algorithm (FH) with parameters that produce at most 143 superpixels per image.

olution $1/32$ in each spatial direction to the original size of the input image. Furthermore, as computing the loss (3) is intractable when considering all possible point-pixel pairs $\mathcal{P} \in \{(\boldsymbol{f}_m^c, \boldsymbol{g}_m^c) : c = 1, \dots, C, m = 1, \dots, M\}$, a random subset of $\mathcal{P}$ is selected to compute it. In practice, as multiple scenes are available in a batch at train time, the set $\mathcal{P}$ also contains the point-pixel pairs of *all* scenes in our implementation. This sampling is not required in our method thanks to the use of superpixels which reduces the number of matching pairs. We use exactly the same parameters for pre-training a network with PPKT† or SLidR.

The noticeable differences between PPKT and PPKT† are the following:

1. the use of cartesian coordinates vs. cylindrical coordinates for f_θ;
2. direct pixel-point correspondences in RGB-D data vs indirect correspondences computed via a projection matrix in autonomous driving;
3. the absence of randomly drop cuboids in PPKT and the absence of random rescaling and elastic distorsion in PPKT†.

C.2. PointContrast

We retrained PointContrast [67] on nuScenes after studying several setups to optimize its performance. PointContrast requires pairs of point clouds acquired from different viewpoints in the same scene with a list of matching points in these two views. To provide a fair baseline, we tested different strategies to create this training dataset. Among the tested strategies, the best one consists in creating all possible pairs of keyframes within a scene, then removing pairs of point clouds which are less than 10 m apart, and removing those which have less than 1024 pairs of matching points.

The list of matching points in a pair of point clouds is computed as follows. We first register both point clouds using the ground truth pose of the Lidar. Then, for each point in one point cloud, we search for the nearest point in the second point cloud and consider that it is a pair of matching points if the points are less than 10 cm apart.

PointContrast† is trained on 1 GPU, with a batch size of 8, using SGD with the same parameters as for SLidR, except for a initial rate set at 1, and cosine annealing scheduler. We selected the learning rate using our mini-val split in order to optimize the performance of PointContrast†. As point cloud augmentations, we used a random rotation around the z-axis, random flip of the x or y-axis and dropped points in cuboids whose sides cover at most 20% of the range of point coordinates in each axis. Finally, one can note that the size of the pre-training dataset is different for PointContrast† and SLidR. For fairness, we set the number of iterations for pre-training with PointContrast† as follows: we compute the total number of point clouds used in SLidR over the 50 training epochs and use the same number of pairs of point clouds in PointContrast†.

C.3. DepthContrast

The last baseline is DepthContrast [72] which pre-trains simultaneously two 3D network backbones, a point-based network, e.g., [51] and a voxel-based network, e.g., [73], using a contrastive task between the global point-cloud representations of the two networks. Among the three baselines, it is the only one which has already been used on a autonomous driving dataset: the Waymo Open Dataset [56]. To make it comparable with the rest of the methods in our study, we re-used the point-based network used in [72] while changing the voxel-based network to the same sparse residual U-Net that processes cylindrical coordinate voxels as in SLidR. After DepthContrast pre-training, we only evaluate on the downstream tasks the voxel-based network.

We trained DepthContrast† on 1 GPU, with a batch size of 8, using SGD with a momentum of 0.9, weight decay of 0.0001, and an initial learning rate of 0.001 (tuned on our mini-val split) that drops to 10^{-6} with a cosine annealing scheduler. We used queues of 60K negatives for the contrastive loss.

D. Additional Training Details

D.1. Linear Probing

In this experiment, the pretrained network f_θ is combined with a pointwise linear classification head which is trained for 50 epochs with a learning rate of 0.05 for all methods.

D.2. Few-Shot Semantic Segmentation

On SemanticKITTI, the networks are fine-tuned for 100 epochs and a batch size of 10. On nuScenes, we use a batch size of 16 and for 100 epochs. We use different learning rates on the classification head and the backbone

Method	barrier	bicycle	bus	car	const. veh.	motorcycle	pedestrian	traffic cone	trailer	truck	driv. surf.	other flat	sidewalk	terrain	manmade	vegetation	**mIoU**
Random	0.0	0.0	8.1	65.0	0.1	6.6	21.0	9.0	9.3	25.8	89.5	14.8	41.7	48.7	72.4	73.3	30.3
PointContrast†	0.0	1.0	5.6	67.4	0.0	3.3	31.6	5.6	12.1	30.8	91.7	21.9	48.4	50.8	75.0	74.6	32.5
DepthContrast†	0.0	0.6	6.5	64.7	0.2	5.1	29.0	9.5	12.1	29.9	90.3	17.8	44.4	49.5	73.5	74.0	31.7
PPKT†	0.0	2.2	20.7	75.4	1.2	13.2	45.6	8.5	17.5	38.4	92.5	19.2	52.3	56.8	80.1	80.9	37.8
SLidR	0.0	3.1	15.2	72.0	0.9	18.8	43.2	12.5	14.7	33.3	92.8	29.4	54.0	61.0	80.2	81.9	38.3

Table 7. Per-class performance on nuScenes using 1% of the annotated scans for fine-tuning. We report the IoU for each class and highlight the best and second best scores with dark blue and light blue backgrounds, respectively.

Method	car	bicycle	motorcycle	truck	other-vehicle	person	bicyclist	motorcyclist	road	parking	sidewalk	other-ground	building	fence	vegetation	trunk	terrain	pole	traffic-sign	**mIoU**
Random	91.2	0.0	9.4	8.0	10.7	21.2	0.0	0.0	89.4	21.4	73.0	1.1	85.3	41.1	84.9	50.1	71.4	55.4	37.6	39.5
PointContrast†	90.1	4.6	5.4	8.1	9.5	21.9	30.8	0.0	90.7	25.6	73.3	0.3	86.4	39.3	83.7	51.2	70.6	53.6	34.9	41.1
DepthContrast†	91.7	8.8	11.5	19.9	15.4	24.7	0.0	0.0	89.5	21.0	72.9	0.7	85.4	40.6	85.1	51.7	71.3	57.9	39.3	41.5
PPKT†	91.3	1.9	11.2	23.1	12.1	27.4	37.3	0.0	91.3	27.0	74.6	0.3	86.5	38.2	85.3	58.2	71.6	57.7	40.1	43.9
SLidR	92.2	3.0	17.0	22.4	14.3	36.0	22.1	0.0	91.3	30.0	74.7	0.2	87.7	41.2	85.0	58.5	70.4	58.3	42.4	44.6

Table 8. Per-class performance on SemanticKITTI using 1% of the annotated scans for fine-tuning. We report the IoU for each class and highlight the best and second best scores with dark blue and light blue backgrounds, respectively.

f_θ, except when the backbone is initialized with random weights. We recall that these learning rates are optimized for each method and each dataset, using our mini-val split for nuScenes and the validation set for semanticKITTI.

D.3. Annotation Efficiency on nuScenes

As in the previous section, we use different learning rates on the classification head and the backbone f_θ and optimize these learning rates for each method and each subset size, using our mini-val split. The network is fine-tuned for 100 epochs when using 1% of annotated data and 50 epochs for the other percentages.

E. nuScenes mini-val Split

The training set of nuScenes contains 700 scenes in total, including:

- 137 raining scenes,
- 84 night-time scenes,
- 310 scenes in Singapore,
- 390 scenes in Boston.

We construct our mini-val split by selecting 100 scenes from this training set so that it contains each type of scenes in the same proportion. Our mini-val split contains:

- 20 raining scenes,
- 12 night-time scenes,
- 44 scenes in Singapore,
- 56 scenes in Boston.

We will provide the list of selected scenes along with the implementation of SLidR.

F. Societal and Environmental Impact

Self-supervision enables the use of large and uncurated datasets with performance often increasing with the duration of the training schedule and the size of the model, at the cost of using much more computational resources with possibly negative environmental impacts. Yet, pre-trained models also reduce the training time needed on multiple downstream tasks, hence reducing the environmental cost of training downstream models. In fact, we distribute our pre-trained models.

G. Public Resources Used

We acknowledge the use of the following public resources, during the course of this work:

- KITTI object detection ..... CC BY-NC-SA 3.0
- MinkowskiEngine ..... MIT License

- nuScenes ..... CC BY-NC-SA 4.0
- nuScenes-devkit ..... Apache License 2.0
- OpenPCDet ..... Apache License 2.0
- Pytorch Lightning ..... Apache License 2.0
- Semantic KITTI ..... CC BY-NC-SA 4.0