

BEVContrast: Self-Supervision in BEV Space for Automotive Lidar Point Clouds

Corentin Sautier^{1,2} Gilles Puy² Alexandre Boulch² Renaud Marlet^{1,2} Vincent Lepetit¹

¹LIGM, Ecole des Ponts, Univ Gustave Eiffel, CNRS, Marne-la-Vallée, France ²Valeo.ai, Paris, France

Abstract

We present a surprisingly simple and efficient method for self-supervision of 3D backbone on automotive Lidar point clouds. We design a contrastive loss between features of Lidar scans captured in the same scene. Several such approaches have been proposed in the literature from PointContrast [40], which uses a contrast at the level of points, to the state-of-the-art TARL [30], which uses a contrast at the level of segments, roughly corresponding to objects. While the former enjoys a great simplicity of implementation, it is surpassed by the latter, which however requires a costly pre-processing. In BEVContrast, we define our contrast at the level of 2D cells in the Bird’s Eye View plane. Resulting cell-level representations offer a good trade-off between the point-level representations exploited in PointContrast and segment-level representations exploited in TARL: we retain the simplicity of PointContrast (cell representations are cheap to compute) while surpassing the performance of TARL in downstream semantic segmentation. The code is available at github.com/valeoai/BEVContrast

1. Introduction

Lidar point cloud processing has recently received a great deal of attention, in particular because of its applicability in the field of autonomous driving. However, annotating a single Lidar point cloud for training a deep architecture can take multiple hours [2], while deep networks in safety-critical autonomous cars will require training on many scans of very diverse scenes, multiplying the need for such annotations. An attractive solution is self-supervised learning, which has been shown to reduce the need for labeled data by pre-training a model on a pretext task requiring no label [6, 14, 25, 35]. Urban scenes are especially suitable for this kind of pre-training, as the cost of obtaining data is only a fraction of the cost of annotating it.

Self-supervised methods based on a contrastive loss are among the best performing approaches. They are constructed on the following principle: deep representations extracted from two views (i.e., two scans) of a scene with sufficient

overlap should be identical. PointContrast [40] uses this principle at the point level, while SegContrast [29] and TARL [30] use it at the level of segments that roughly correspond to objects. After identifying pairs of corresponding points or segments, the contrastive loss forces (i) the representations in each pair to be as similar as possible while (ii) keeping representations in two different pairs as dissimilar as possible. An illustration of the application of this contrastive principle at the point and segment levels is available in Figure 1.

The advantage of considering point-level representations as in PointContrast is the simplicity of the method: these representations are readily available at the output of the 3D backbone. Experiments nevertheless show that point-level representations do not generalize as well as segment-level representations to semantic downstream tasks [29, 30]. Unfortunately, the latter require a complex segmentation step as pre-processing, with the addition of several hyperparameters to tune. For example, TARL requires temporal aggregation of multiple scans, careful road plane detection to remove ground points, and point clustering on the remaining points to create point cloud segments to contrast.

In this work, we propose to contrast features at the level of 2D cells of a grid on the Bird’s Eye View (BEV) plane. The feature of a cell is here defined as the average of the features of the points projecting into that cell. We call the resulting method BEVContrast. It is motivated by the fact that objects in urban scenes are naturally well separated in the BEV plane. Therefore, locally averaging point features in this plane permits us to obtain cell-level features, which are a good trade-off between point-level representations, and segment-level representations. The high-level principle of BEVContrast is illustrated in Figure 1. It retains the simplicity of PointContrast (the projection in BEV and local average pooling in each BEV cell being cheap to compute) while experiments show that it competes with the best self-supervised methods such as TARL, despite the fact that we do not treat dynamic objects explicitly.

Our contributions are the following. First, we propose a novel self-supervised method for Lidar point clouds that retains the simplicity of PointContrast while surpassing all con-

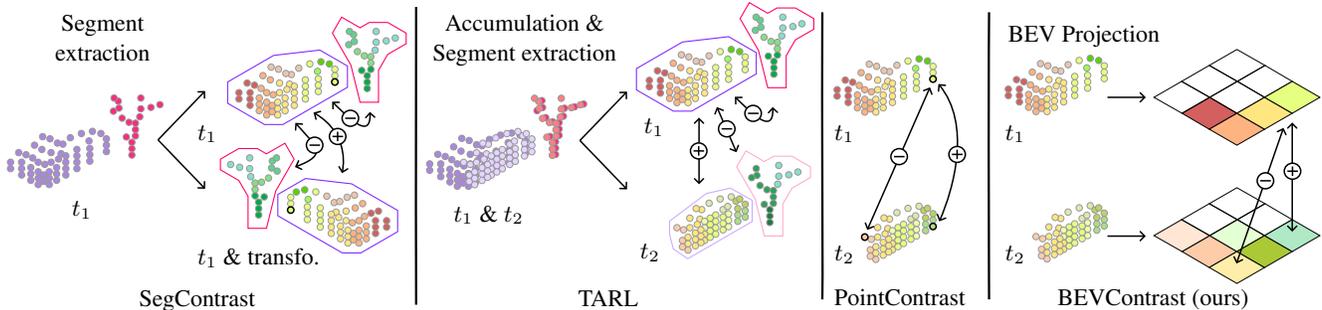


Figure 1. **Comparison of several self-supervised contrastive methods with BEVContrast.** SegContrast [29] and TARL [30] learn powerful representations by contrasting features at the level of segments obtained thanks to a careful and costly pre-processing. PointContrast [40] is simpler as it requires no pre-processing and directly contrasts representations at the point level, but experiments show that these point-level representations are not as powerful as segment-level representations. BEVContrast preserves the simplicity of PointContrast by extracting cell-level representations in BEV. These representations are cheap to compute, require no pre-processing and perform better than SegContrast and TARL for downstream semantic segmentation. (\oplus and \ominus represent positive and negative contrastive losses respectively; t_1 and t_2 stand for two different time steps.)

current self-supervised methods for downstream semantic segmentation, including non-contrastive methods too, such as ALSO [3]. Second, we show that projecting and pooling features in bird’s eye views is surprisingly more effective than pooling features over segments extracted with complex methods. Finally, as BEVContrast works in bird’s eye views, it can be applied out of the box to pre-train backbones for 3D object detection such as SECOND [41] or PVRCNN [36], with which we achieve competing results compared to the state-of-the-art.

2. Related work

Self-supervision consists in learning representations by creating a pretext task from the data itself without the need of any annotations. The goal is to obtain good representations that generalize well to many downstream tasks where limited annotations are available. In this section, we provide an overview of self-supervised training methods working on images, point clouds, or both modalities.

Self-supervision on images. Early self-supervised methods defined pretext tasks such as predicting by which amount an image was rotated [12] or solving a jigsaw puzzle [28]. Today, the most successful class of self-supervised methods either leverage discriminative tasks [5–7, 13–15] in which a network is trained to extract representations that are invariant to augmentations or a masked image modeling task [1, 16] in which the network is trained to reconstruct image parts hidden from the network input.

Self-supervision on point clouds. The advances on self-supervision on point clouds followed closely the improvements made on images. Early self-supervised methods were using pretext tasks such as predicting transformation applied on the point cloud or reconstructing parts of the point

cloud [33, 34]. These methods were applied on dense scans of single objects.

The advent of discriminative based methods permitted the development of efficient self-supervised methods working on entire indoor or outdoor scenes. PointContrast [40], DepthContrast [46] and STRL [18] were the among the first such methods. Concerning methods targeting specifically automotive Lidar data, ProposalContrast [43], SegContrast [29] and TARL [30] construct a discriminative tasks by pooling features at the level of segments that roughly correspond to objects. The three methods first filter out ground points, e.g., using RANSAC [9] for SegConstant or Patchwork [21] for TARL, and then extract segments either as a spherical region [43] or using HDBSCAN [26]. In addition, TARL leverages the temporal dimension and aggregates multiple scans to generate temporally-consistent segments. The temporal axis is also used in STSSL [39], which combines point-level contrastive learning within each segment and segment-level contrastive learning between frames. In contrast, our method shows that simply projecting and pooling features in BEV is surprisingly more effective than pooling over such segments.

Finally, reconstruction-based methods are also successful for self-supervision on point clouds. Masked image modeling tasks have also been adapted to point clouds data. Some methods reconstruct points coordinates using the Chamfer distance [17, 32, 42, 45], other predict voxel occupancy [27], or predict point-patch token from a codebook [10, 44]. Recently, ALSO [3] proposes to use unsupervised surface reconstruction as a pretext task to train 3D backbones on automotive Lidar point clouds. Using the knowledge of occupancy before and after an observed 3D point along a Lidar line of sight, it learns to construct an implicit occupancy function and, incidentally, good point features. Although it does not exploit contrast, it reaches high performance on downstream

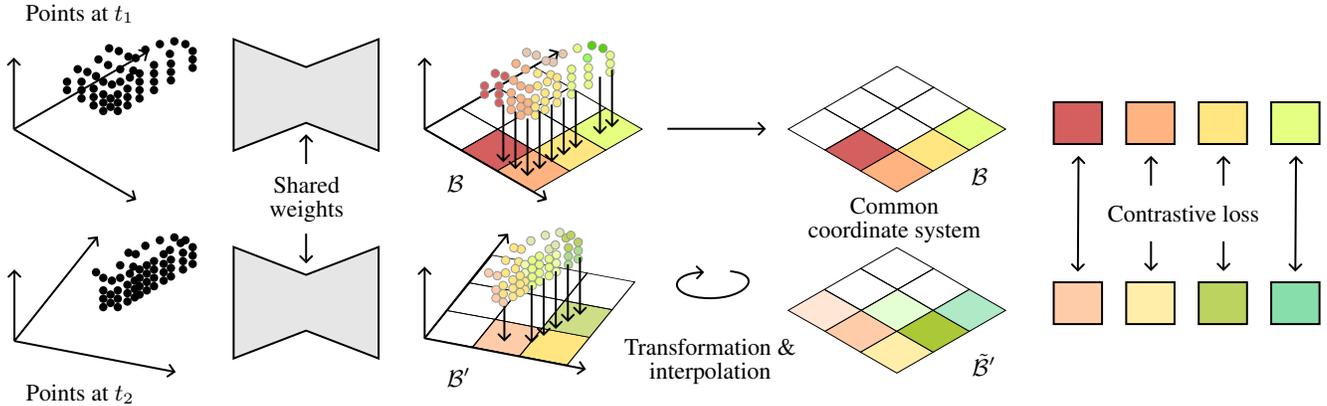


Figure 2. **Overview of BEVContrast.** Two point cloud views of the same scenes are encoded and projected on the BEV plane. One BEV is aligned to the other’s coordinate frame by affine transformation and interpolation, and a loss constrains the features in both BEV views.

semantic segmentation and object detection.

Multi-modal self-supervision. Another line of work leverages synchronized and calibrated cameras and Lidar to pre-train a 3D backbone [19, 22, 24, 35]. The underlying idea is to find pairs of corresponding points and pixels and ensure that the associated point and pixel representations are as close as possible. We note that the image backbone has to be pre-trained on an external dataset, e.g., ImageNet [8], for [22, 24, 35]. While these methods can obtain impressive results, the use of another modality as well as an external image dataset makes them not comparable to our method.

3. Method

Our method (BEVContrast) is designed to pre-train a 3D backbone $f_\theta(\cdot)$ from Lidar scans without any annotations, where θ represents the learnable parameters. We describe below the method for 3D segmentation backbones, i.e., when $f_\theta(\cdot)$ takes as input a point cloud \mathcal{P} and outputs a D -dimensional feature vector for each point. We denote the feature vector for the i^{th} point in \mathcal{P} by $f_\theta(\mathcal{P})_i \in \mathbb{R}^D$. However, the method is also usable for 3D object detection with minor modifications, as described in Section 3.4.

3.1. Pre-training Data

BEVContrast requires pairs of partially overlapping Lidar scans ($\mathcal{P}, \mathcal{P}'$) captured in a same scene but from different viewpoints. Typically, these point clouds will be captured at different instants separated by a few seconds Δ_{time} . Each point of \mathcal{P} or \mathcal{P}' is described by a 4-dimensional vector containing its 3D Cartesian xyz -coordinates and the measured return intensity (reflectance).

We also need the 3D rotation matrix $R \in \mathbb{R}^{3 \times 3}$ and 3D translation vector $t \in \mathbb{R}^3$ that register \mathcal{P}' to \mathcal{P} . R and t can be computed from the absolute or relative poses of the Lidar, which are usually available in autonomous driving datasets.

3.2. BEV pooling

As discussed in the introduction, PointContrast enjoys a great simplicity of implementation by working directly on point-level representations. However, these learned representations do not rival with segment-level representations such as those in SegContrast and TARL [29, 30]. Yet, the higher performance of SegContrast and TARL, compared to PointContrast, comes at the cost of an expensive pre-processing that requires extra hyperparameters to tune.

In this work, instead of pooling point representations over expensive-to-get segments, we propose to project features on the BEV plane and locally pool features over 2D cells of a grid defined on this plane. We show in the next section that this simple and fast-to-compute pooling mechanism is key to reach state-of-the-art results.

Our pooling mechanism is constructed on the fact that most of the objects in a Lidar scan are naturally well separated in the BEV plane. We thus propose to project the points in \mathcal{P} in the BEV plane of height $z = 0$, divide this plane in regular cells of size $b \times b$, and average the representations in $f_\theta(\mathcal{P})$ of points falling in a same BEV cell. Thanks to this process, and for a reasonable cell size b , each BEV cell feature describes an object or parts of an object with little “contamination” from neighboring objects. These cell-level features are thus a good proxy for object-level representations. Mathematically, we denote these BEV representations $\mathcal{B} = g(f_\theta(\mathcal{P}), \mathcal{P}) \in \mathbb{R}^{M \times M \times D}$ and $\mathcal{B}' = g(f_\theta(\mathcal{P}'), \mathcal{P}') \in \mathbb{R}^{M \times M \times D}$, where $M \times M$ is the number of BEV cells, and $g(\cdot, \cdot)$ represents the projection into BEV and average pooling step. Cells with no point projected into them are filled with null features.

3.3. Contrastive loss

We construct our self-supervised contrastive loss directly on the BEV representations. The first step is to align the BEV

representations \mathcal{B} and \mathcal{B}' to put cells in correspondences. In practice, we exploit the information contained in R and t to compute a 2D affine transformation which we use to register \mathcal{B}' onto \mathcal{B} thanks to a bilinear interpolation. We denote by $\tilde{\mathcal{B}}'$ the feature map \mathcal{B}' transformed into the same BEV grid as \mathcal{B} . We provide the details about this affine transformation in the supplementary material, and a study of alternative registration methods in Table 6.

We pre-train $f_\theta(\cdot)$ using a discriminative task at the level of BEV cells. Let us denote $\tilde{\mathcal{B}}'_l \in \mathbb{R}^D$ and $\mathcal{B}_m \in \mathbb{R}^D$ the D -dimensional representations extracted from $\tilde{\mathcal{B}}'$ and \mathcal{B} in the l^{th} and m^{th} BEV cells, respectively. Our loss enforces BEV representations \mathcal{B}_l and $\tilde{\mathcal{B}}'_l$ falling into the same cell l to be identical, while keeping BEV representations \mathcal{B}_m and $\tilde{\mathcal{B}}'_l$ falling in two different cells l and m as different as possible. Concretely, we sample at random a set \mathcal{N} of non-empty BEV cells and minimize

$$\mathcal{L}(\theta) = - \sum_{l \in \mathcal{N}} \log \left[\frac{\exp(\tilde{\mathcal{B}}'_l \cdot \mathcal{B}_l / \tau)}{\sum_{m \in \mathcal{N}} \exp(\tilde{\mathcal{B}}'_l \cdot \mathcal{B}_m / \tau)} \right], \quad (1)$$

where $\tau > 0$ is a temperature hyperparameter and \cdot denotes the scalar product in \mathbb{R}^D . Note that the BEV cell features are ℓ_2 -normalized before computing the above loss.

3.4. Adaptation to 3D detection

Note that BEVContrast can be applied out of the box to pre-train popular object detection backbones such as those used in SECOND [41] or PVRCNN [36]. Indeed, these backbones output, by design, features in a BEV plane grid, hence, we do not need to apply any projection. In this case, the BEV pooling and choice of b is omitted as it is taken care of by the backbone, while the affine transformation parameters are obtained as above.

4. Experiments

In this section, we compare BEVContrast to state-of-the-art methods for self-supervision. We compare the performance of these methods using two downstream tasks: semantic segmentation in Section 4.2 and object detection in Section 4.3. Then, we study the sensitivity of BEVContrast to its hyperparameters in Section 4.4. We conclude in Section 4.5 with a discussion about the advantages and disadvantages of BEVContrast and the concurrent methods.

4.1. Datasets

nuScenes (NS) [4] contains 700 and 150 scenes for training and validation, acquired in Boston and Singapore with a 32-beam Lidar. For a fair comparison with the related work, we pre-train our network using 600 training scenes and exploit the remaining 100 training scenes to tune all hyperparameters. The performance of all methods are compared on

the official validation set of nuScenes, which thus plays the role of the test set. Note that nuScenes sequences contain unannotated scans that could be used for self-supervision: scans are captured at 20 Hz but in fact only annotated at 2 Hz. However, to maintain comparability with previous work, even during pre-training, we only use the scans that are annotated, although we ignore the annotations. The pre-trained backbones are then fine-tuned on different subsets of the training scenes. One can refer to [3, 24, 35] for more details about this protocol, which we follow exactly.

SemanticKITTI (SK) [2] contains 10 training sequences and 1 validation sequence captured with a 64-beam Lidar in various environments in the vicinity of Karlsruhe in Germany. We exploit the full training set for pre-training and the partial training sets defined in [29] for the downstream tasks.

SemanticPOSS (SP) [31] is composed of 6 sequences captured on the campus of the Peking University with a 40-beam Lidar and annotated with a subset of the labels classes of SemanticKITTI. We use this dataset in Section 4.5 to test the capacity of self-supervised pre-trained backbones in generalizing to different Lidars. We fine-tune the pre-trained backbones using the partial training sets used in [3] and evaluate the performance on the official validation set.

KITTI 3D Detection (K) [11] consists in 7.5k Lidar frames split into a training set and a validation set. The annotations are 3D bounding boxes around cars, bicyclists and pedestrians. Unlike the previously described datasets, KITTI3D does not contain any information that allows one to reconstruct a temporal sequence of scans. This prevents us to pre-train any backbone on this dataset, but we can use it for downstream object detection.

4.2. Semantic segmentation

In this section, we pre-train 3D backbones for semantic segmentation on either nuScenes or SemanticKITTI, and evaluate the quality of the pre-training by downstream fine-tuning on subsets of nuScenes or SemanticKITTI. Each of these subsets contains either 0.1%, 1%, 10%, 50% or 100% of the annotated training data. All self-supervised pre-trained backbones are fine-tuned using the protocol of [3]. For methods not already evaluated in [3], we obtained the pre-trained backbones directly from the authors and run the fine-tuning experiments ourselves.

3D backbones. We follow the common practice in the literature (see, e.g., [3]) and pre-train a ResUNet18 on SemanticKITTI and a ResUNet34 on nuScenes.

BEVContrast pre-training protocol. We pre-train the backbones with BEVContrast for 50 epochs with a batch size of 10, split across two A100-40GB GPUs for SemanticKITTI and a batch size of 24 on a single A100 for

Dataset	Method	0.1%		1%		10%		50%		100%	
nuScenes	No pre-training	21.6	±0.5	35.0	±0.3	57.3	±0.4	69.0	±0.2	71.2	±0.2
	PointContrast [†] [40]	27.1	±0.5	37.0	±0.5	58.9	±0.2	69.4	±0.3	71.1	±0.2
	DepthContrast [†] [46]	21.7	±0.3	34.6	±0.5	57.4	±0.5	69.2	±0.3	71.2	±0.2
	ALSO [3]	26.2	±0.5	37.4	±0.3	59.0	±0.4	69.8	±0.2	71.8	±0.2
	BEVContrast (ours)	26.6	±0.5	37.9	±0.4	59.0	±0.6	70.5	±0.2	72.2	±0.1
SemanticKITTI	No pre-training	30.0	±0.2	46.2	±0.6	57.6	±0.9	61.8	±0.4	62.7	±0.3
	PointContrast [‡] [40]	32.4	±0.5	47.9	±0.5	59.7	±0.5	62.7	±0.3	63.4	±0.4
	SegContrast [29]	32.3	±0.3	48.9	±0.3	58.7	±0.5	62.1	±0.4	62.3	±0.4
	DepthContrast [†] [46]	32.5	±0.4	49.0	±0.4	60.3	±0.5	62.9	±0.5	63.9	±0.4
	STSSL [39]	32.0	±0.4	49.4	±1.1	60.0	±0.6	62.9	±0.7	63.3	±0.3
	ALSO [3]	35.0	±0.1	50.0	±0.4	60.5	±0.1	63.4	±0.5	63.6	±0.5
	TARL [30]	37.9	±0.4	52.5	±0.5	61.2	±0.3	63.4	±0.2	63.7	±0.3
	BEVContrast (ours)	39.7	±0.9	53.8	±1.0	61.4	±0.4	63.4	±0.6	64.1	±0.4

[†] as reimplemented by ALSO [3]

[‡] as reimplemented by SegContrast [29]

Table 1. **Semantic segmentation fine-tuning results on nuScenes and SemanticKITTI** using different subsets of the corresponding training set for fine-tuning (from 0.1% to 100%). The 3D backbones are pre-trained and fine-tuned on the same datasets. A single pre-training is used for each line, and details about pre-training data are given in Section 4.1. We report the average and standard deviation of the mIoU% over 5 different fine-tunings. Individual classwise fine-tuning results are in the supplementary material.

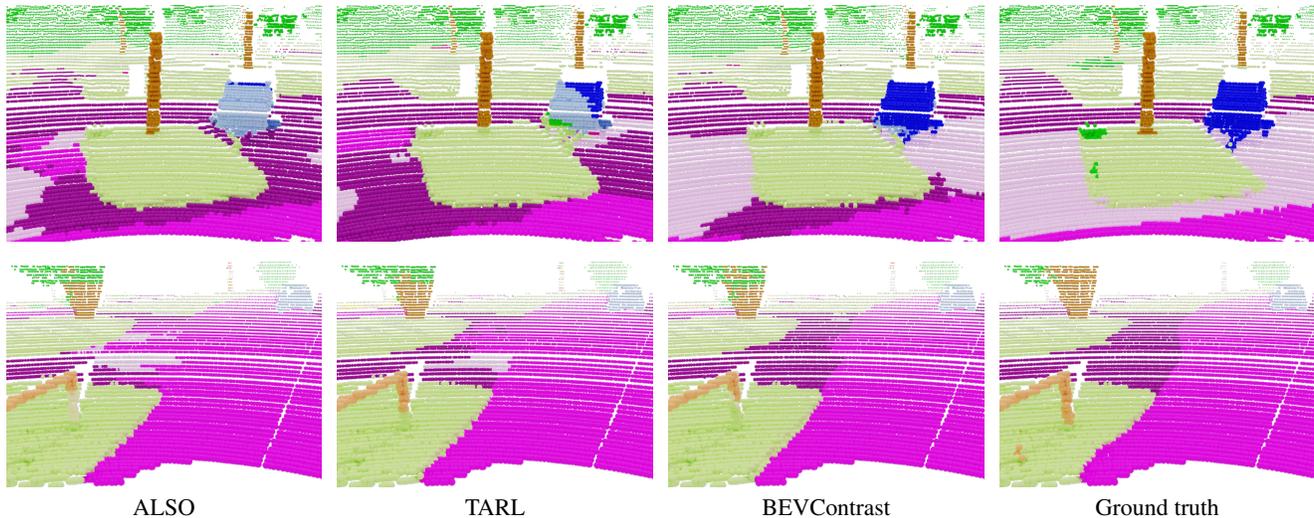


Figure 3. **Semantic segmentation visualizations on SemanticKITTI** after pre-training on the full training dataset and fine-tuning using 1% of the annotated data. car, other veh., sidewalk, drivable surf., parking, terrain, vegetation, trunk, fence

nuScenes. The pre-training uses the full training set of SemanticKITTI, and 600 training scenes out of the 700 available for nuScenes (see [3] for details). We use AdamW [23] with a learning rate of 0.001, a weight decay of 0.001, $\mathcal{N} = 4096$ samples in the loss, a temperature hyperparameter τ of 0.07 as in [15, 40] and a cosine annealing scheduler. The hyperparameters b and Δ_{time} have been set to 30 cm and 1 s on nuScenes and 20 cm and 0.7 s on SemanticKITTI. We limit the size M of the BEV to 512 for SemanticKITTI and 256 for nuScenes.

Fine-tuning protocol. After pre-training, the backbones are

fine-tuned for semantic segmentation on different subsets of SemanticKITTI or nuScenes. We use the fine-tuning hyperparameters and splits defined in [3]. Specifically, we use a batch size of 8 for nuScenes, and 2 for SemanticKITTI. We compute the per-point final score by setting each point’s prediction as the voxel prediction it fall in. For each pre-trained model and each subset, we perform 5 independent fine-tunings and report the average and standard deviation of the mIoU over these runs.

Other details. All experiments on SemanticKITTI (pre-training and fine-tuning) in [3, 29, 30] are done by filtering

Method	Data.	Cars	Ped.	Cycl.	mAP	Diff.
SECOND - R_{40} metric						
No pre-training	-	81.5	48.8	65.7	65.4	
ALSO [3]	NS	81.8	54.2	68.2	68.1	+2.7
BEVContrast (ours)	NS	81.4	51.9	69.3	67.6	+2.2
SECOND - R_{11} metric						
No pre-training	-	78.6	53.0	67.2	66.3	
Voxel-MAE [27]	K	78.9	53.1	68.1	66.7	+0.4
ALSO [3]	NS	78.7	55.2	68.1	67.3	+1.0
BEVContrast (ours)	NS	78.3	52.8	69.3	66.8	+0.5
PV-RCNN - R_{40} metric						
No pre-training	-	84.5	57.1	70.1	70.6	
STRL [18]	K	84.7	57.8	71.9	71.5	+0.9
GCC-3D [20]	NS	-	-	-	70.8	+0.2
GCC-3D [20]	W	-	-	-	71.3	+0.7
PointCont. [40]	W	84.2	57.7	72.7	71.6	+1.0
Prop.Cont. [43]	W	84.7	60.4	73.7	72.9	+2.3
ALSO [3]	NS	84.9	57.8	75.0	72.5	+1.9
BEVContrast (ours)	NS	84.8	57.3	74.2	72.1	+1.5
PV-RCNN - R_{11} metric						
No pre-training	-	83.6	57.9	70.5	70.7	
Voxel-MAE [27]	K	83.8	59.4	72.0	71.7	+1.0
ALSO [3]	NS	83.8	58.5	74.4	72.2	+1.5
BEVContrast (ours)	NS	83.6	58.0	73.1	71.6	+0.9

Table 2. **Detection results on KITTI3D [11]**, validation set, moderate difficulty. We report AP (%) and the dataset used for pre-training each method.

out the few points labelled as “ignore.” In our method, we use all points during pre-training, as no labels should be used during this stage, but filter out these points during fine-tuning. Hence, all fine-tuning experiments are done in the same setting as the above methods.

Results. We compare in Table 1 the performance of several pre-training methods. We observe that BEVContrast is the state-of-the-art method on all datasets and all splits in the case where the backbones are pre-trained and fine-tuned on the same datasets. Surprisingly, per-class results (Table 8 in the appendix) do not show a significant difference between static and mobile objects, even though TARL’s segmentation should be robust to them. We repeat the observation that contrastive methods based on segment pooling are better performing than point-level contrast. We also present qualitative results obtained with different methods in Figure 3.

4.3. Detection

In this section, we evaluate the interest of using our method to pre-train two widely-used object detectors: SECOND [41] and PVRCNN [36]. We compare BEVContrast to methods which already pre-train these object detectors and for which results are available in the literature. Note that these object

detectors are *not* pre-trained in TARL [30] and, as it is not straightforward to adapt TARL for this backbone, we do not report any comparison with TARL for object detection.

Backbone. We pre-train the backbone which is shared by the object detectors SECOND and PVRCNN. This backbone is made of a 3D sparse encoder, which outputs a BEV representation of the input point cloud, and is followed by a 2D encoder, which refines the BEV representation.

BEVContrast pre-training protocol. We pre-train this backbone with BEVContrast for 50 epochs with a batch size of 10 on a single A100-40GB GPUs on the 600 training scenes of nuScenes reserved for pre-training. We use AdamW [23] with a learning rate of 0.001, a weight decay of 0.001, a temperature τ of 0.07 and a cosine annealing scheduler. As we target downstream fine-tuning on KITTI3D, we use the OpenPCDet voxel size tuned for this dataset: 5 cm on the x and y axes, 10 cm on the z axis. We limit the range to a square of side 102.4 m centered on the ego-car during pre-training, as done in OpenPCDet when training on nuScenes. We do not use any extra pooling after the 2D encoder: the parameter b is fixed by the BEV resolution and the convolution strides chosen in SECOND [41] and PVRCNN [36].

Fine-tuning protocol. After pre-training, the detection head of SECOND or PVRCNN is appended to the pre-trained backbone and the whole network is fine-tuned on KITTI3D. We use the OpenPCDet [38] implementation of these object detectors and the default OpenPCDet training parameters. As in [3], we perform 3 independent fine-tunings and report the best mAP on the validation set of KITTI3D.

Results. We present the results obtained with BEVContrast and concurrent methods in Table 2. First, we notice that pre-training with BEVContrast always leads to better results than without any pre-training at all. Second, when comparing methods that are also pre-trained on nuScenes, we notice that BEVContrast performs better than GCC-3D and competes with ALSO, while being significantly simpler to implement. Finally, when considering all methods and all pre-training datasets, BEVContrast is in the top 3 of all methods. Please note that this experiment involves a single run for each method, rather than averaging the performance across multiple runs. The error margin may thus be significant, especially as the variance is generally higher for object detection than for semantic segmentation. Nevertheless, contrastive methods, including ours, seem slightly less effective on object detection compared to semantic segmentation. This could arise from the nature of detection, which is a more geometrical task, compared to segmentation, which is a per-point classification task, as contrastive learning tends not to encourage the learning of geometric object positioning, contrary to reconstruction-based methods such as ALSO.

	nuScenes		
	$b = 20$ cm	$b = 30$ cm	$b = 40$ cm
$\Delta_{\text{time}} = 0.5$ s	58.1	57.4	56.6
$\Delta_{\text{time}} = 1.0$ s	58.2	58.7	57.2
$\Delta_{\text{time}} = 1.5$ s	58.5	57.1	56.7
$\Delta_{\text{time}} = 2.0$ s	58.3	58.3	57.6

Table 3. **Sensitivity to Δ_{time}** , the time difference between the capture of \mathcal{P} and \mathcal{P}' , and b , the size of the BEV cells for pooling. The backbone is pre-trained on 600 scenes extracted from the original training set of nuScenes, and fine-tuned on 60 scenes. We report the mIoU% on the spared 100 scenes from the original training set of nuScenes.

4.4. Sensitivity to hyperparameters

In this section, we study the sensitivity of BEVContrast to its hyperparameters. We use the following experimental protocol. All models are pre-trained for 20 epochs on either nuScenes or SemanticKITTI with a batch size of 2. The models are then fine-tuned on their pre-training datasets exactly as described in Section 4.2, except for SemanticKITTI where the batch size is set to 4. We use the splits with 10% of annotations for fine-tuning.

4.4.1 Sensitivity to Δ_{time} and b

We start by studying the sensitivity of our method to the choice of the hyperparameters Δ_{time} , the time difference between the acquisitions of \mathcal{P} and \mathcal{P}' , and b , the size of the BEV cells.

We present the results obtained when pre-training and fine-tuning on nuScenes in Table 3. We do not notice dramatic drop of performance for any of tested pairs $(\Delta_{\text{time}}, b)$. The best results are obtained at $b = 30$ cm and $\Delta_{\text{time}} = 1.0$ s, which are the parameters we used to produce the results in Table 1.

The results obtained when pre-training and fine-tuning on SemanticKITTI are presented in Table 4. The set of tested parameters is smaller than for nuScenes, but we still notice that the results are relatively stable at $b = 20$ cm for all tested Δ_{time} , with a variation of at most 1.0 points of mIoU. The results presented in Table 1 were obtained by using $b = 20$ cm and $\Delta_{\text{time}} = 0.7$ s.

4.4.2 Choice of the scans to contrast

Many different strategies exist to select overlapping point clouds \mathcal{P} and \mathcal{P}' in BEVContrast. We study two simple alternatives in this section: selecting scans acquired Δ_{time} seconds apart, or selecting scans acquired after a displacement of (at least) Δ_{dist} meters of the ego-car.

	SemanticKITTI		
	$b = 10$ cm	$b = 20$ cm	$b = 40$ cm
$\Delta_{\text{time}} = 0.5$ s	-	60.3	59.9
$\Delta_{\text{time}} = 0.7$ s	58.8	61.3	60.4
$\Delta_{\text{time}} = 0.9$ s	-	60.7	-

Table 4. **Sensitivity to Δ_{time}** , the time difference between the capture of \mathcal{P} and \mathcal{P}' , and b , the size of the BEV cells for pooling. The backbone is pre-trained on the full training set of SemanticKITTI, and fine-tuned on 10% of the training set. We report the mIoU% on the validation set of SemanticKITTI.

	Δ_{time}			Δ_{dist}			
	0.5 s	0.7 s	0.9 s	5 m	10 m	15 m	20 m
	60.3	61.3	60.7	60.0	60.5	60.6	59.6

Table 5. **Effect on the performance when selecting the point clouds \mathcal{P} and \mathcal{P}' acquired Δ_{time} seconds apart or Δ_{dist} meters apart.** The results are obtained by pre-training, with $b = 20$ cm, and fine-tuning on SemanticKITTI. We report the mIoU% on the validation set of SemanticKITTI.

We conduct this study on SemanticKITTI with $b = 20$ cm and report the results in Table 5. We notice that the performance is rather robust whether we select scans based on the time of acquisition or the displacement of the ego-car. On SemanticKITTI, we achieve the best results with $\Delta_{\text{time}} = 0.7$ s. For experiments on the other datasets, we privileged a selection based on Δ_{time} , as it also easier to implement as sequences are acquired with a fixed frame rate.

4.4.3 Study of different registration methods

BEVContrast requires aligned BEV representations for pre-training the 3D backbone $f_{\theta}(\cdot)$. In this section, we study different strategies to obtain these aligned representations:

1. The first option, which we denote by “3D,” consists in registering \mathcal{P}' after computing $f_{\theta}(\mathcal{P}')$ but before projection in BEV. We compute $g(f_{\theta}(\mathcal{P}'), r(\mathcal{P}'))$ where $r(\cdot)$ applies the rigid 3D transformation given by R and t .
2. The second option consists in aligning the representations after projection in BEV, as proposed in Section 3, but using a nearest neighbor interpolation instead of bilinear interpolation. We denote this option by “2D NN.”
3. The third option consists in the alignment described in Section 3, i.e., using a bilinear interpolation. We denote this option by “2D Bi.”

Note that “2D NN” and “2D Bi.” can be used when pre-training object detection backbones, but not “3D” as these backbones often output directly a BEV representation where height information is lost.

The results are presented in Table 6. We notice that 2D

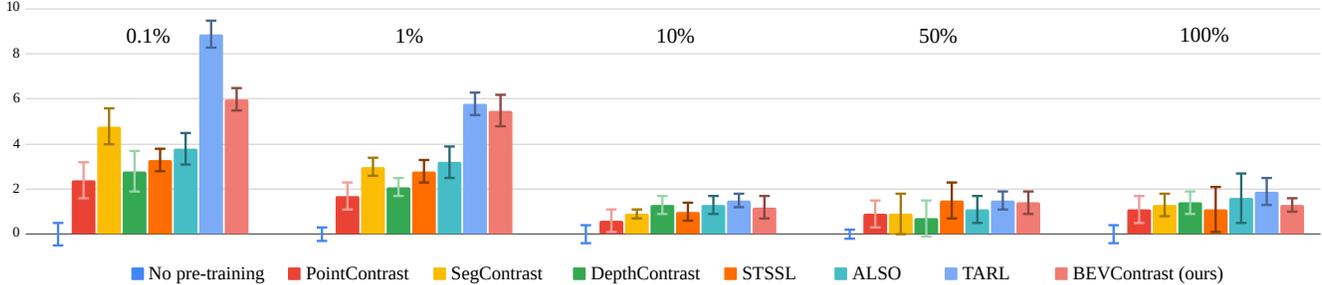


Figure 4. **Semantic segmentation fine-tuning results on SemanticPOSS** after pre-training on SemanticKITTI. Different subsets from 0.1% to 100% of the training set of SemanticPOSS are used for fine-tuning. We report the relative improvement compare to *no pre-training* in mIoU%, average and standard deviation over 5 different fine-tunings. The individual fine-tuning results are in the supplementary material.

$2D Bi.$	$2D NN$	$3D$
61.3	59.5	60.0

Table 6. **Effect of the registration method.** We analyze the impact of three registration methods to align the representation B and B' . $2D Bi.$ and $2D NN$ are aligning the representations after projection in BEV, the first using a bilinear interpolation and the second using nearest neighbor interpolation. $3D$ is aligning the representation before projection in BEV. The backbones are pre-trained on SemanticKITTI and fine-tuned on the same dataset using 10% of annotated scans with $\Delta_{time} = 0.7$ s and $b = 20$ cm. We report the mIoU% on the validation set of SemanticKITTI.

registration with bilinear interpolation leads to much better results than with nearest neighbor interpolation. Interestingly as well, 3D registration does not perform as well as 2D registration with bilinear interpolation. We hypothesize that approximate registration of the BEV representations with “ $2D Bi.$ ” acts as a form of regularizer which avoids overfitting and leads to 3D representations that generalize better to the downstream task.

4.5. Limitation and Discussion

Transfer between datasets. We study the edge case of generalization to data acquired with a Lidar different from the one used for the pre-training data. Let us mention nevertheless that, in the context of self-supervision, this setting is mostly relevant in the rare case where one is unable to collect sufficient data for self-supervision in the setting of interest. We consider SemanticKITTI as the pre-training dataset and fine-tune the pre-trained backbone on SemanticPOSS. The results are presented in Figure 4. We observe that BEVContrast achieves competitive results and is ranked second or third on every annotated portion of the training set of SemanticPOSS. Furthermore, it is in the error margin with the SOTA on all but the 0.1% split, as when using more annotated scans, the ranking of the methods becomes much less clear as the variability of the results is large.

Sequential data. STSSL [39], TARL [30] and BEVContrast, and more generally all methods that use two or more frames, require the datasets to be provided as sequential data. It is the case for most datasets [2, 4, 25, 37], but not for KITTI detection benchmark, where frame timestamps are not provided. This prevents us from to pre-train on KITTI. In comparison, other methods such as ALSO [3] and VoxelMAE [27] process one frame at a time, allowing them to pre-train on any dataset, but making them unable to exploit the redundancy of sequential acquisition.

Pre-processing. An advantage of BEVContrast is the absence of pre-processing before pre-training while, e.g., the pre-processing in TARL costs about 45 s/frame on a single core of a Xeon 4114 CPU on scans of SemanticKITTI. Furthermore, the hyperparameters for road detection and segment extraction in [29, 30] are numerous (minimum cluster size, condition to merge different cluster, etc.) and difficult to tune when changing the type of Lidar, e.g., from the 64-beam Lidar in SemanticKITTI to the 32-beam Lidar in nuScenes or 40-beam Lidar in SemanticPOSS. Instead, for BEVContrast, the sensitivity study shows that choosing $b \approx 20$ cm and $\Delta_{time} \approx 1$ s is not far from optimal on both nuScenes and SemanticKITTI.

False negatives and object segmentation. In an ideal setting, one would like to contrast features at the level of actual objects, but objects are unknown in the annotation-less self-supervision scenario, and have somehow to be approximated. In practice, BEV cells in BEVContrast do not always align well on objects: large objects are over-split into several cells, while a single cell may sometimes contain more than one actual object. Undersegmentation and oversegmentation occur similarly with the segmentation methods of SegContrast and TARL. Fortunately, contrastive approaches are somewhat robust to such point grouping errors, yielding nevertheless good features. Our finding is that, surprisingly, our cell-level point grouping, despite its simplicity, apparent coarseness, and absence of filtering of dynamic objects, is actually more effective than complex segmentation methods that try

to discover objects. It even outperforms TARL in downstream fine-tuning on both nuScenes and SemanticKITTI in all tested scenarios, although TARL uses time information across successive scans to produce better segments.

5. Conclusion

We presented BEVContrast, a simple yet effective contrastive self-supervised method to pre-train 3D backbones for automotive Lidar point clouds. It conveys the surprising observation that contrasting representations of cells on a regular BEV grid performs on par with the most sophisticated contrastive methods, that resort to unsupervised clusterization algorithms. With our method, we are able to reach SOTA results on self-supervised semantic segmentation on nuScenes and SemanticKITTI, while demonstrating the competitiveness of our method on object detection on KITTI [11] with a different backbone.

Acknowledgements: This work was performed using HPC resources from GENCI-IDRIS (Grants 2023-AD011013765). This work was supported in part by the French Agence Nationale de la Recherche (ANR) grant MultiTrans (ANR21-CE23-0032)

References

- [1] Hangbo Bao, Li Dong, and Furu Wei. BEiT: BERT pre-training of image transformers. In *ICLR*, 2021. 2
- [2] 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*, 2019. 1, 4, 8
- [3] Alexandre Boulch, Corentin Sautier, Björn Michele, Gilles Puy, and Renaud Marlet. ALSO: Automotive lidar self-supervision by occupancy estimation. In *CVPR*, 2023. 2, 4, 5, 6, 8, 1, 3
- [4] 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*, 2020. 4, 8
- [5] Mathilde Caron, Ishan Misra, J. Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *NeurIPS*, 2020. 2
- [6] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021. 1
- [7] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geofrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*. PMLR, 2020. 2
- [8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009. 3
- [9] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 2
- [10] Kexue Fu, Mingzhi Yuan, Shaolei Liu, and Manning Wang. Boosting point-bert by multi-choice tokens. *TCSVT*, 2023. 2
- [11] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *CVPR*, 2012. 4, 6, 9
- [12] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018. 2
- [13] 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*, 2021. 2
- [14] Jean-Bastien Grill, Florian Strub, Florent Alché, 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*, 2020. 1
- [15] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020. 2, 5
- [16] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022. 2
- [17] Georg Hess, Johan Jaxing, Elias Svensson, David Hagerman, Christoffer Petersson, and Lennart Svensson. Masked autoencoder for self-supervised pre-training on lidar point clouds. In *WACV*, 2023. 2
- [18] Siyuan Huang, Yichen Xie, Song-Chun Zhu, and Yixin Zhu. Spatio-temporal self-supervised representation learning for 3D point clouds. In *ICCV*, 2021. 2, 6
- [19] Zhenyu Li, Zehui Chen, Ang Li, Liangji Fang, Qinhong Jiang, Xianming Liu, Junjun Jiang, Bolei Zhou, and Hang Zhao. SimIPU: Simple 2D image and 3D point cloud unsupervised pre-training for spatial-aware visual representations. In *AAAI*, 2022. 3
- [20] 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*, 2021. 6
- [21] Hyungtae Lim, Oh Minho, and Hyun Myung. Patchwork: Concentric zone-based region-wise ground segmentation with ground likelihood estimation using a 3d lidar sensor. *IEEE Robotics and Automation Letters*, 2021. 2
- [22] Yueh-Cheng Liu, Yu-Kai Huang, Hung-Yueh 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
- [23] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 5, 6
- [24] Anas Mahmoud, Jordan SK Hu, Tianshu Kuai, Ali Harakeh, Liam Paull, and Steven L Waslander. Self-supervised image-to-point distillation via semantically tolerant contrastive loss. In *CVPR*, 2023. 3, 4

- [25] Jiageng Mao, Minzhe Niu, Chenhan Jiang, hanxue liang, Jingheng Chen, Xiaodan Liang, Yamin Li, Chaoqiang Ye, Wei Zhang, Zhenguo Li, Jie Yu, Hang Xu, and Chunjing Xu. One million scenes for autonomous driving: ONCE dataset. In *NeurIPS Datasets and Benchmarks Track (Round 1)*, 2021. 1, 8
- [26] Leland McInnes, John Healy, and Steve Astels. hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software*, 2(11):205, 2017. 2
- [27] Chen Min, Dawei Zhao, Liang Xiao, Yiming Nie, and Bin Dai. Occupancy-MAE: Self-supervised pre-training large-scale lidar point clouds with masked occupancy autoencoders. *arXiv:2206.09900*, 2023. 2, 6, 8
- [28] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016. 2
- [29] Lucas Nunes, Rodrigo Marcuzzi, Xieyuanli Chen, Jens Behley, and Cyrill Stachniss. Segcontrast: 3D point cloud feature representation learning through self-supervised segment discrimination. *IEEE RA-L*, 7(2):2116–2123, 2022. 1, 2, 3, 4, 5, 8
- [30] Lucas Nunes, Louis Wiesmann, Rodrigo Marcuzzi, Xieyuanli Chen, Jens Behley, and Cyrill Stachniss. Temporal consistent 3D lidar representation learning for semantic perception in autonomous driving. In *CVPR*, 2023. 1, 2, 3, 5, 6, 8, 4
- [31] Yancheng Pan, Biao Gao, Jilin Mei, Sibao Geng, Chengkun Li, and Huijing Zhao. Semanticpos: A point cloud dataset with large quantity of dynamic instances, 2020. 4
- [32] Yatian Pang, Wenxiao Wang, Francis EH Tay, Wei Liu, Yonghong Tian, and Li Yuan. Masked autoencoders for point cloud self-supervised learning. In *ECCV*, 2022. 2
- [33] Omid Poursaeed, Tianxing Jiang, Quintessa Qiao, Nayun Xu, and Vladimir G. Kim. Self-supervised learning of point clouds via orientation estimation. In *3DV*, 2020. 2
- [34] Jonathan Sauder and Bjarne Sievers. Self-supervised deep learning on point clouds by reconstructing space. In *NeurIPS*, 2019. 2
- [35] Corentin Sautier, Gilles Puy, Spyros Gidaris, Alexandre Boulch, Andrei Bursuc, and Renaud Marlet. Image-to-lidar self-supervised distillation for autonomous driving data. In *CVPR*, 2022. 1, 3, 4
- [36] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. PV-RCNN: Point-voxel feature set abstraction for 3D object detection. In *CVPR*, 2020. 2, 4, 6, 1
- [37] Pei Sun, Henrik Kretschmar, 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*, 2020. 8
- [38] OpenPCDet Development Team. OpenPCDet: An open-source toolbox for 3D object detection from point clouds. <https://github.com/open-mmlab/OpenPCDet>, 2020. 6
- [39] Yanhao Wu, Tong Zhang, Wei Ke, Sabine Süsstrunk, and Mathieu Salzmann. Spatiotemporal self-supervised learning for point clouds in the wild. In *CVPR*, 2023. 2, 5, 8, 3, 4
- [40] 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*, 2020. 1, 2, 5, 6
- [41] Yan Yan, Yuxing Mao, and Bo Li. SECOND: Sparsely embedded convolutional detection. *Sensors*, 18, 2018. 2, 4, 6, 1
- [42] Honghui Yang, Tong He, Jiaheng Liu, Hua Chen, Boxi Wu, Binbin Lin, Xiaofei He, and Wanli Ouyang. GD-MAE: generative decoder for MAE pre-training on lidar point clouds. In *CVPR*, 2023. 2
- [43] Junbo Yin, Dingfu Zhou, Liangjun Zhang, Jin Fang, Cheng-Zhong Xu, Jianbing Shen, and Wenguan Wang. Proposal-Contrast: Unsupervised pre-training for lidar-based 3D object detection. In *ECCV*, 2022. 2, 6
- [44] Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-bert: Pre-training 3d point cloud transformers with masked point modeling. In *CVPR*, 2022. 2
- [45] Renrui Zhang, Ziyu Guo, Peng Gao, Rongyao Fang, Bin Zhao, Dong Wang, Yu Qiao, and Hongsheng Li. Point-M2AE: multi-scale masked autoencoders for hierarchical point cloud pre-training. *NeurIPS*, 2022. 2
- [46] Zaiwei Zhang, Rohit Girdhar, Armand Joulin, and Ishan Misra. Self-supervised pretraining of 3D features on any point-cloud. In *ICCV*, 2021. 2, 5

BEVContrast: Self-Supervision in BEV Space for Automotive Lidar Point Clouds

Supplementary Material

A. Alignment of the BEV representations

A.1. Registration

Our loss is constructed by aligning the BEV representations \mathcal{B} and \mathcal{B}' .

One possibility to align these representations is to register \mathcal{P}' onto \mathcal{P} after computing $f_\theta(\mathcal{P}')$ but before projection in BEV, i.e., computing $g(f_\theta(\mathcal{P}'), r(\mathcal{P}'))$ (see section 4.4.3). For each point p'_i with coordinates (x'_i, y'_i, z'_i) in \mathcal{P}' , the corresponding transformed coordinates $r(\mathcal{P}')_i = (X'_i, Y'_i, Z'_i)$ satisfies

$$\begin{aligned} \begin{pmatrix} X'_i \\ Y'_i \\ Z'_i \end{pmatrix} &= \begin{pmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{pmatrix} \begin{pmatrix} x'_i \\ y'_i \\ z'_i \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} \quad (2) \\ &= \begin{pmatrix} R_{11}x'_i + R_{12}y'_i + R_{13}z'_i + t_1 \\ R_{21}x'_i + R_{22}y'_i + R_{23}z'_i + t_2 \\ R_{31}x'_i + R_{32}y'_i + R_{33}z'_i + t_3 \end{pmatrix}, \quad (3) \end{aligned}$$

where we used the rotation matrix R and translation vector t defined in Section 3.1. The function $g(\cdot, \cdot)$ then averages the representation of the points whose coordinates (X'_i, Y'_i) falls in the same BEV cell. Unfortunately, our experiments (see Section 4.4.3) show that this approach leads to suboptimal results.

The approach we advocate for consists in projecting and pooling the representations before alignment, i.e., computing $g(f_\theta(\mathcal{P}'), \mathcal{P}')$ (where the representations of the points whose coordinates (x'_i, y'_i) fall in the same BEV cell are averaged) and then aligning the representation by bilinear interpolation. As the projection in BEV makes us loose all access about the actual value of z'_i , we approximate the value of (X'_i, Y'_i) in (3) by (X''_i, Y''_i) where

$$\begin{pmatrix} X''_i \\ Y''_i \end{pmatrix} = \begin{pmatrix} R_{11}x'_i + R_{12}y'_i + t_1 \\ R_{21}x'_i + R_{22}y'_i + t_2 \end{pmatrix}. \quad (4)$$

This approximation holds when $|R_{13}z'_i| \ll |R_{11}x'_i + R_{12}y'_i + t_1|$ and $|R_{23}z'_i| \ll |R_{21}x'_i + R_{22}y'_i + t_2|$, e.g., when there are only small rotations around the x and y -axis (often observed in practice on our datasets) or for all points whose coordinates satisfy $z'_i \approx 0$. Furthermore, this formulation allows easy adaptation to backbones with outputs in the BEV plane, such as those of Second [41] and PVRCNN [36].

A.2. Bilinear interpolation details.

Empty cells in the BEV representation \mathcal{B}' are zero-valued, and so are padding cells at the border of the grid. The interpolation to obtain $\tilde{\mathcal{B}}'$ from \mathcal{B}' being bilinear, the features in each cell after transformation will be a weighted sum of

features before the transformation, some of which might be zero vectors. To avoid the influence of the varying number of empty cells around cell filled cells during interpolation, one might want to rescale (cell-wise) the interpolated BEV representation $\tilde{\mathcal{B}}'$ to take into account the number of empty cells involved in the interpolation. This cell-wise rescaling, which would increase the complexity of implementation, is fortunately not necessary because the interpolated feature in each cell of $\tilde{\mathcal{B}}'$ are ℓ_2 -normalized immediately after for computing the loss.

B. Detailed results and visualizations

We present in Table 10 and Table 9 the per class IoUs obtained on SemanticKITTI and nuScenes, respectively, for BEVContrast and others when available. On SemanticKITTI, BEVContrast performs better for most classes.

For completeness, we report in Table 9, Table 10, Table 11 the mIoU of each of the five individual runs performed to obtain the main results in the paper.

Figure 5 illustrates the better segmentation quality on the class other vehicles of BEVContrast compared to ALSO [3] and TARL [30].

Finally, we present in Table 12 all object detection metrics automatically computed in OpenPCDet on KITTI3D for our method, ALSO and no pretraining. The metric use the Average Precision with 40 recall positions where a correct prediction is a box with an overlap of at least 70% for a car and 50% for a pedestrian or a cyclist. The precisions are provided in the 2D image plane, BEV horizontal plane, as well as using the 3D coordinates. The “orientation similarity” corresponds to the Average Orientation Similarity for the correct prediction. It is defined as the average over all correct predictions of $(1 + \cos(\delta))/2$ where δ is the difference between the predicted and ground truth orientations for any matching boxes.

Method	<i>mIoU</i>	<i>barrier</i>	<i>bicycle</i>	<i>bus</i>	<i>car</i>	<i>const. veh.</i>	<i>motorcycle</i>	<i>pedestrian</i>	<i>traffic cone</i>	<i>trailer</i>	<i>truck</i>	<i>driv. surf.</i>	<i>other flat</i>	<i>sidewalk</i>	<i>terrain</i>	<i>manmade</i>	<i>vegetation</i>
BEVContrast	37.9	0.0	1.3	32.6	74.3	1.1	0.9	41.3	8.1	24.1	40.9	89.8	36.2	44.0	52.1	79.9	79.7

Table 7. Per-class IoU on nuScenes using 1% of the annotated scans for fine-tuning. Results are averaged over 5 different fine-tunings. Only BEVContrast per-class results can be given as previous methods did not detail those information.

Method	<i>mIoU</i>	<i>car</i>	<i>bicycle</i>	<i>motorcycle</i>	<i>truck</i>	<i>other-veh.</i>	<i>person</i>	<i>bicyclist</i>	<i>motorcycli.</i>	<i>road</i>	<i>parking</i>	<i>sidewalk</i>	<i>oth. ground</i>	<i>building</i>	<i>fence</i>	<i>vegetation</i>	<i>trunk</i>	<i>terrain</i>	<i>pole</i>	<i>traffic sign</i>
STSSL [39]	49.4	92.8	8.3	27.2	32.0	25.3	48.3	57.7	0.0	90.6	31.2	73.9	0.7	87.7	43.3	86.6	57.1	72.4	57.9	46.2
TARL [30]	52.5	93.4	16.6	45.1	43.5	21.7	49.4	61.8	0.1	91.0	33.4	75.1	0.5	89.0	43.8	87.5	61.7	74.0	59.9	50.3
BEVContrast	53.8	93.8	15.6	39.0	50.1	28.9	52.5	61.8	0.2	92.0	40.1	76.1	0.5	90.4	49.6	87.7	61.1	73.7	60.6	48.5

Table 8. Per-class IoU% on SemanticKITTI using 1% of the annotated scans for fine-tuning. Results are averaged over 5 different fine-tunings.

%	Method	Runs					Average and std
0.1%	No pre-training	21.88	21.21	22.05	21.08	21.96	21.64 ±0.45
	ALSO [3]	26.62	26.86	25.99	25.59	26.08	26.23 ±0.51
	BEVContrast	27.25	26.47	26.21	26.06	26.76	26.55 ±0.47
1%	No pre-training	34.86	35.09	34.72	34.72	35.55	34.99 ±0.35
	ALSO [3]	37.42	37.52	37.15	37.11	37.94	37.43 ±0.34
	BEVContrast	37.66	38.28	38.16	37.41	38.01	37.90 ±0.36
10%	No pre-training	57.62	57.66	57.31	56.70	57.19	57.30 ±0.39
	ALSO [3]	58.63	58.62	59.11	59.28	59.35	59.00 ±0.35
	BEVContrast	58.29	58.66	58.74	59.77	59.58	59.01 ±0.64
50%	No pre-training	68.80	68.90	68.94	69.31	69.01	68.99 ±0.19
	ALSO [3]	69.69	69.58	69.93	69.66	70.17	69.81 ±0.24
	BEVContrast	70.14	70.47	70.80	70.59	70.59	70.52 ±0.24
100 %	No pre-training	71.21	71.35	71.20	70.93	71.32	71.20 ±0.17
	ALSO [3]	71.95	71.92	71.60	71.88	71.39	71.75 ±0.24
	BEVContrast	72.17	72.18	71.97	72.11	72.34	72.15 ±0.13

Table 9. Details of individual run on nuScenes semantic segmentation. We report the mIoU% on the official validation set for each of the individual runs averaged in the main paper.

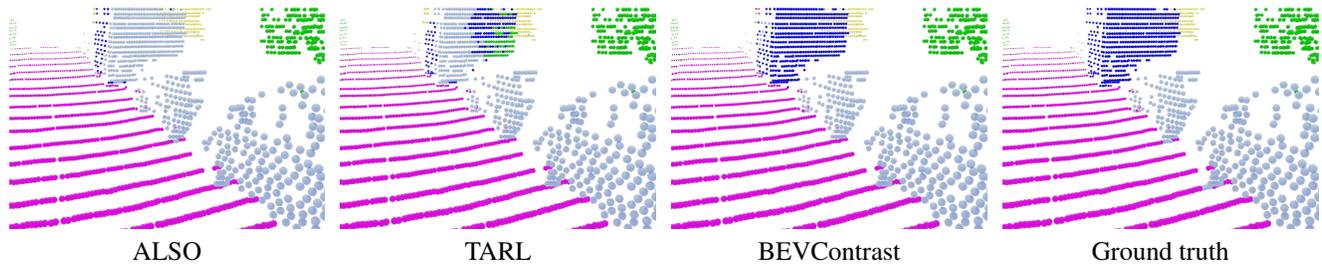


Figure 5. **Semantic segmentation visualizations on SemanticKITTI** after pre-training on the full training dataset and fine-tuning using 1% of the annotated data. ■ *car*, ■ *other veh.*, ■ *drivable surf.*, ■ *vegetation*

%	Method	Runs					Average and std
0.1%	No pre-training	30.22	29.99	29.74	30.15	29.77	29.97 \pm 0.22
	STSSL [39]	31.37	32.62	32.07	32.03	32.01	32.02 \pm 0.44
	ALSO [3]	34.97	34.83	34.81	35.10	35.11	34.96 \pm 0.14
	TARL [30]	38.01	38.25	37.89	37.87	37.28	37.86 \pm 0.36
	BEVContrast	39.08	40.28	38.42	40.53	40.31	39.72 \pm 0.92
1%	No pre-training	45.1	46.32	46.59	46.68	46.49	46.24 \pm 0.65
	STSSL [39]	50.03	48.90	51.00	48.22	49.00	49.43 \pm 1.09
	ALSO [3]	50.04	50.28	49.43	50.41	50.02	50.04 \pm 0.38
	TARL [30]	52.09	52.41	52.35	53.43	52.32	52.52 \pm 0.52
	BEVContrast	53.59	54.28	53.86	54.92	52.35	53.80 \pm 0.95
10%	No pre-training	57.04	58.74	57.71	56.27	58.01	57.55 \pm 0.94
	STSSL [39]	60.09	59.19	60.65	60.29	59.69	59.98 \pm 0.56
	ALSO [3]	60.41	60.45	60.47	60.54	60.43	60.46 \pm 0.05
	TARL [30]	61.04	60.92	61.64	60.97	61.19	61.15 \pm 0.29
	BEVContrast	61.82	60.74	61.74	61.28	61.31	61.38 \pm 0.43
50%	No pre-training	61.48	62.33	61.88	61.80	61.31	61.76 \pm 0.39
	STSSL [39]	63.50	63.33	61.91	62.54	63.29	62.91 \pm 0.67
	ALSO [3]	63.09	63.43	62.99	63.28	64.15	63.39 \pm 0.46
	TARL [30]	63.23	63.66	63.43	63.25	63.27	63.37 \pm 0.18
	BEVContrast	62.43	63.27	64.07	63.77	63.65	63.44 \pm 0.63
100 %	No pre-training	62.49	62.35	62.98	62.50	63.06	62.68 \pm 0.32
	STSSL [39]	63.27	63.50	63.33	62.74	63.66	63.30 \pm 0.35
	ALSO [3]	64.29	63.75	63.75	63.34	63.07	63.64 \pm 0.46
	TARL [30]	63.83	63.71	63.14	63.90	63.92	63.70 \pm 0.32
	BEVContrast	63.94	63.84	64.05	64.74	63.73	64.06 \pm 0.40

Table 10. Details of individual run on SemanticKITTI semantic segmentation. We report the mIoU% on the official validation set for each of the individual runs averaged in the main paper.

%	Method	Runs					Average and std
0.1%	No pre-training	37.23	37.53	36.37	36.72	36.59	36.89 \pm 0.48
	STSSL [39]	41.04	39.97	40.05	39.78	40.33	40.23 \pm 0.49
	ALSO [3]	40.04	41.23	41.29	40.88	39.83	40.65 \pm 0.68
	TARL [30]	45.38	46.33	45.29	45.40	46.61	45.80 \pm 0.62
	BEVContrast	43.06	42.92	42.03	43.25	43.04	42.86 \pm 0.48
1%	No pre-training	46.99	46.23	46.09	46.33	46.47	46.42 \pm 0.35
	STSSL [39]	48.86	48.58	49.65	49.55	49.57	49.24 \pm 0.49
	ALSO [3]	50.55	48.85	49.02	49.86	49.51	49.56 \pm 0.68
	TARL [30]	52.24	52.34	52.11	52.85	51.45	52.20 \pm 0.50
	BEVContrast	50.91	52.13	52.88	52.03	51.48	51.89 \pm 0.74
10%	No pre-training	54.29	53.96	54.53	54.95	54.60	54.47 \pm 0.37
	STSSL [39]	55.03	55.10	55.96	55.84	55.43	55.47 \pm 0.42
	ALSO [3]	56.16	56.19	55.43	55.15	56.14	55.81 \pm 0.49
	TARL [30]	56.29	55.52	56.02	56.12	55.95	55.98 \pm 0.29
	BEVContrast	55.10	55.92	55.50	55.12	55.60	55.45 \pm 0.35
50%	No pre-training	55.48	55.19	55.40	55.27	55.04	55.28 \pm 0.17
	STSSL [39]	55.62	57.96	56.68	57.01	56.72	56.80 \pm 0.84
	ALSO [3]	57.07	55.56	56.71	56.13	56.30	56.35 \pm 0.58
	TARL [30]	55.97	56.86	56.98	56.86	57.09	56.75 \pm 0.45
	BEVContrast	56.65	55.84	56.96	57.21	57.01	56.73 \pm 0.54
100 %	No pre-training	54.52	55.52	55.52	55.10	54.83	55.10 \pm 0.44
	STSSL [39]	54.71	56.34	56.56	57.48	56.09	56.24 \pm 1.00
	ALSO [3]	56.88	58.23	55.45	56.01	56.88	56.69 \pm 1.05
	TARL [30]	56.70	56.42	57.92	57.01	56.94	57.00 \pm 0.56
	BEVContrast	56.31	56.50	56.83	56.21	55.92	56.35 \pm 0.34

Table 11. Details of individual run on SemanticPOSS semantic segmentation. We report the mIoU% on the official validation set with a pre-training on SemanticKITTI for each of the individual runs averaged in the main paper.

Backbone	Metric	Method	Car			Pedestrian			Cyclist			
			Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	
SECOND	2D object detection	Scratch [†]	95.84	94.49	92.00	68.27	64.69	61.15	91.02	78.88	76.00	
		ALSO [3]	97.48	94.70	93.56	72.39	68.69	65.86	91.34	81.64	78.46	
		BEVContrast	97.45	94.53	93.42	71.93	67.33	63.84	93.61	80.85	77.30	
	Bird's eye view	Scratch [†]	93.76	89.82	87.65	59.74	54.85	50.56	87.19	70.96	68.00	
		ALSO [3]	94.64	90.77	88.24	64.32	59.13	55.25	86.92	74.58	70.17	
		BEVContrast	92.33	89.79	87.47	61.45	56.46	52.74	90.64	72.86	68.35	
	3D object detection	Scratch [†]	90.20	81.50	78.61	53.89	48.82	44.56	82.59	65.72	62.99	
		ALSO [3]	90.21	81.78	78.97	59.56	54.24	50.27	81.12	68.19	64.10	
		BEVContrast	90.18	81.44	78.60	56.60	51.95	47.64	87.35	69.28	65.06	
	Orientation similarity	Scratch [†]	95.83	94.35	91.79	63.70	59.52	55.85	90.86	78.44	75.52	
		ALSO [3]	97.45	94.54	93.30	67.67	63.33	60.28	91.01	80.75	77.49	
		BEVContrast	97.42	94.42	93.22	66.86	61.80	58.16	93.36	79.67	76.15	
	PV-RCNN	2D object detection	Scratch [†]	97.86	94.39	93.92	73.84	68.68	65.53	94.34	81.89	77.36
			ALSO [3]	96.12	94.45	93.99	73.70	68.70	65.31	94.61	81.86	78.67
			BEVContrast	97.55	94.40	94.04	72.65	67.55	64.77	96.42	82.74	79.70
Bird's eye view		Scratch [†]	94.65	90.61	88.56	68.28	60.62	55.95	92.52	75.03	70.40	
		ALSO [3]	93.10	90.64	88.53	68.72	60.92	56.96	93.11	76.74	73.06	
		BEVContrast	92.79	90.61	88.49	68.01	61.40	57.49	93.70	74.96	71.22	
3D object detection		Scratch [†]	91.74	84.60	82.29	65.51	57.49	52.71	91.37	71.51	66.98	
		ALSO [3]	92.31	84.86	82.61	65.60	57.76	52.96	91.70	74.98	70.67	
		BEVContrast	91.48	84.47	82.37	64.85	58.21	53.36	90.90	71.80	68.02	
Orientation similarity		Scratch [†]	97.84	94.25	93.70	69.73	63.89	60.31	94.20	81.00	76.47	
		ALSO [3]	96.09	94.29	93.76	67.66	62.74	59.37	94.23	81.07	77.86	
		BEVContrast	97.52	94.27	93.83	67.26	61.99	58.98	96.16	82.14	79.11	

Table 12. Details of object detection metrics on KITTI3D with a pre-training on nuScenes, for all difficulties. Details of the metrics are provided above. The results presented in the main paper use the “3D object detection” metric.