ALSO: Automotive Lidar Self-supervision by Occupancy estimation

Alexandre Boulch, Corentin Sautier, Björn Michele, Gilles Puy, Renaud Marlet

ALSO: Automotive Lidar Self-supervision by Occupancy estimation

Alexandre Boulch[1] Corentin Sautier[1,2] Björn Michele[1,3] Gilles Puy[1] Renaud Marlet[1,2]

[1]Valeo.ai, Paris, France [2]LIGM, Ecole des Ponts, Univ Gustave Eiffel, CNRS, Marne-la-Vallée, France
[3]CNRS, IRISA, Univ. Bretagne Sud, Vannes, France

Abstract

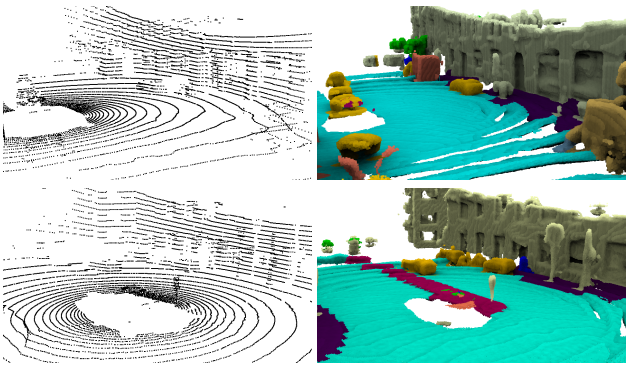
We propose a new self-supervised method for pre-training the backbone of deep perception models operating on point clouds. The core idea is to train the model on a pretext task which is the reconstruction of the surface on which the 3D points are sampled, and to use the underlying latent vectors as input to the perception head. The intuition is that if the network is able to reconstruct the scene surface, given only sparse input points, then it probably also captures some fragments of semantic information, that can be used to boost an actual perception task. This principle has a very simple formulation, which makes it both easy to implement and widely applicable to a large range of 3D sensors and deep networks performing semantic segmentation or object detection. In fact, it supports a single-stream pipeline, as opposed to most contrastive learning approaches, allowing training on limited resources. We conducted extensive experiments on various autonomous driving datasets, involving very different kinds of lidars, for both semantic segmentation and object detection. The results show the effectiveness of our method to learn useful representations without any annotation, compared to existing approaches.

The code is available at github.com/valeoai/ALSO

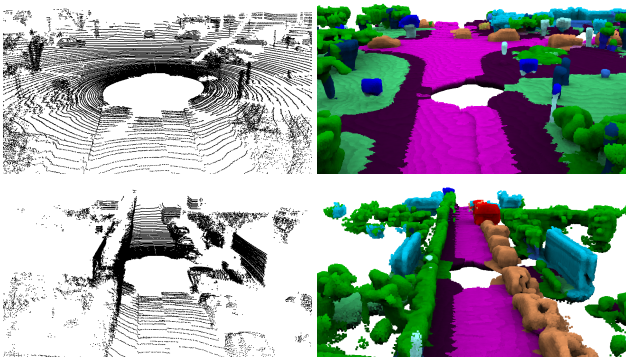
1. Introduction

As a complement to 2D cameras, lidars directly capture the 3D environment of a vehicle with high accuracy and low sensitivity to adverse conditions, such as low illumination, bright sunlight or oncoming headlights. They are thus essential sensors for safe autonomous driving.

Most state-of-the-art lidar-based perception methods, whether they regard semantic segmentation [21, 76, 94] or object detection [44, 73, 87, 90], assume they can be trained on large annotated datasets. However, annotating 3D data for such tasks is notoriously costly and time consuming. As data acquisition is much cheaper than data annotation, being able to leverage unannotated data to increase the performance or reduce the annotation effort is a significant asset.



(a) nuScenes.



(b) SemanticKITTI.

Figure 1. Aggregation of the self-supervised training on lidar datasets. Input point cloud (first column) and occupancy prediction colored by the learned downstream labels.

A promising direction to address this question is to pre-train a neural network using only unannotated data, e.g., on a pretext task which does not require manual labelling, and then to fine-tune the resulting self-supervised pre-trained network for the targeted downstream task(s). With adequate pre-training, the learned network weights are a good starting point for further supervised optimization; training a specific downstream task then typically requires fewer annotations to reach the same performance level as if trained from scratch.

A number of self-supervised approaches have been very

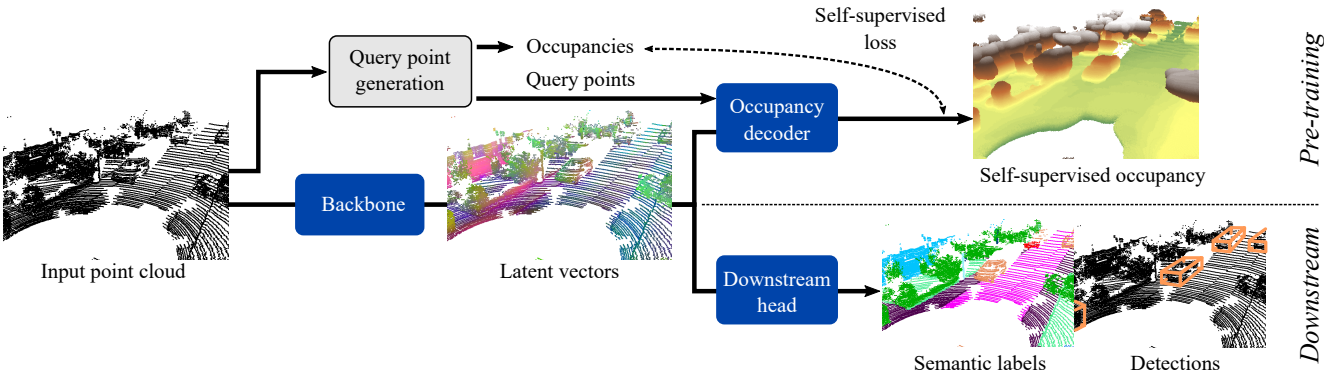


Figure 2. Overview of the approach. The backbone to pre-train produces latent vectors for each input point. At pre-training time, the latent vector are fed into an volumetric occupancy head that classifies query points as full or empty. At semantic training or test time, the same latent vectors are fed into a semantic head, e.g., for semantic segmentation or object detection.

successful in 2D (images), even reaching the level of supervised pre-training [12, 16, 32, 36]. Some self-supervised ideas have been proposed for 3D data as well, which are often transpositions in 3D of 2D methods [39, 58]. Most of them focus on contrastive learning [64, 72, 86, 89, 93] which learns to infer perceptual features that are analogous for similar objects while being far apart for dissimilar objects.

Only few such methods apply to lidar point clouds, which have the particularity of having very heterogeneous densities.

In this work, we propose a totally new pretext task for the self-supervised pre-training of neural networks operating on point clouds. We observe that one of the main reasons why downstream tasks may fail is related to the sparsity of data. Indeed, with automotive lidars, 3D points are especially sparse when far from the sensor or on areas where laser beams have a high incidence on the scanned surface. In such cases, objects are difficult to recognize, and even more so if they are small, such as so-called vulnerable road user (e.g., pedestrians, bicyclists) and traffic signs.

In a mostly supervised context, geometric information such as object shape [42, 61] and visibility information [38] have proved to boost detection performance. Our approach uses visibility-based surface reconstruction as a pretext task for self-supervision. It takes root in the implicit shape representation literature, where shapes are encoded into latent vectors, that can be decoded into a function indicating the shape volume occupancy or the distance to the shape surface. The intuitive idea is that if a network is able to properly reconstruct the 3D geometry of a scene from point clouds, then there are good chances that it constructs rich features that can be reused in a number of other contexts, in particular regarding semantic-related tasks.

Our contributions are as follows: (1) we combine surface reconstruction and visibility information to create a sensor-agnostic and backbone-agnostic pretext task on 3D point clouds, which produces good self-supervised point features for semantic segmentation and object detection; (2) we design a loss that leads each point to capture enough knowledge to reconstruct its neighborhood (instead of aggregating information from neighbors for a more accurate surface reconstruction), which instils a taste of semantics in the geometric task; (3) based on experiments across seven datasets, our self-supervised features, that require only limited resources for training (single 16G GPU), outperform state-of-the-art self-supervised features on semantic segmentation, and are on par with these features on object detection.

2. Related work

Self-supervision has been the subject of a significant research effort, including for image applications. Early methods mostly focused on direct estimation of the transformation applied to images [22, 30, 60, 63, 92]. More recently a great boost of performances has been accomplished with contrastive learning [15, 59, 78, 85], clustering based methods [10, 11] and reconstruction-based approaches. The latter can operate the reconstruction in the feature space [12, 16, 28, 29, 32] trying to reconstruct the features issued from a teacher signal or in the images domain [4, 36], a partially masked input image being reconstructed.

2.1. 2D adaptation to point clouds

2D methods have being adapted to point cloud, in particular for detection. In the dataset presentation paper of ONCE [54], the authors produce self-supervision baselines using methods adapted from image self-supervision: BYOL [32], SWaV [11] and DeepCluster [10]. Another work, related to MAE [36] (images) or Point-MAE [66] (part segmentation): Voxel-MAE [58] reconstruct the complete voxel grid, given a partially masked input.

Our approach is a reconstruction approach. The major difference lies in our supervision signal which is not made by masking an already sparse input, but by estimating the unknown underlying scene surface using sensor information.

2.2. Self-supervision for point clouds.

Classification and part segmentation. Following the same trends as image methods, pretext tasks have been built in order to reconstruct the input point cloud [71, 79], estimate a global transformation [17, 69], contrast between objects views [14, 23, 70, 81] or estimate clusters [35, 91].

Semantic segmentation. Scene level pre-training has been tackled using multi-temporal data, for example, by contrasting point-wise representation which are matched across two temporally distinct and registered acquisitions of the same scene [37, 43, 86]. Segcontrast [64] extract segments likely to belong to the same object (ground plane using RANSAC [26], other cluster using DBSCAN [25]) and then contrast between segment representations for different augmentation of the same scene. DepthContrast [93] contrast four representations obtained with two networks and two augmented views of the same scene. The method is shown to be efficient for indoor data as well as outdoor data, but requires a joint-training of two networks, thus is memory intensive. In STRL [39], a model and an exponential moving average version of it are fed two temporally-close point cloud frames, altered with various 3D augmentations, and the objective is that both representation are similar.

Another class of methods explores cross-modality, leveraging one or multiple images [45, 49, 72]. In our case, we condider that only lidar modality is available.

Our method is, as opposed to all previously cited approaches, not a contrastive method. We do not rely on several augmentations of the scenes, processed in parallel to build our representations. Therefore, our approach can be easily trained on a single 16GB memory GPU.

Detection. Several methods have been adapted from semantic segmentation to detection. PointContrast [86], DepthContrast [93] or STRL [39] propose an extension to detection by training the 3D backbone of the detectors [73, 87].

The current best performing methods are specially designed for 3D detection. GCC3D [46] uses both a contrastive and clustering mechanisms to learn a 3D object detection encoder. Augmented versions of a scene are encoded and a local contrastive loss is applied to enforce feature invariance. Feature learning is then refined with a clustering objective using temporal clues. ProposalContrast [89] applies contrastive learning at region level arguing that scene-level representation may lose details while point-level contrast favors small receptive field, without object-level knowledge.

Our approach also achieves object-size knowledge, but we only require a single parameter, which is the size of the neighborhood to be reconstructed, which intuitively should have a similar dimension to objects in the scene. Moreover, our method can be applied indifferently for semantic segmentation or detection, contrarily to the best performing methods [46, 58, 89].

2.3. Occupancy reconstruction

Surface reconstruction is a well studied subject in computational geometry. Surfaces are usually described either using explicit representations (voxels [55, 83], surface point clouds [1, 48, 88] or meshes [31, 34, 47, 53, 62, 80]) or with implicit representations, which define a function over the 3D space from which the surface can be extracted.

Implicit reconstruction with deep networks. Implicit representations have gained in popularity since the seminal work DeepSDF [67]. The existing methods estimate either a distance function [20, 33, 57, 67] (signed or not), an occupancy function [6, 18, 56] or both [24].

Our approach predicts an occupancy function, i.e., label the 3D space as inside or outside the volume.

Global and local representations. Surface reconstruction from point clouds has mainly been tackled with two distinct objectives. On the one hand, shape representation [9, 56, 67] aims at associating each object with a single latent vector containing rich global geometric information, in a space suitable for interpolation, possibly with good classification properties [7]. On the other hand, surface reconstruction using local representation [6, 19, 20, 24, 40, 68, 82] aims at precise surface estimation and is able to scale to large scenes. However, the local description focuses on low-level geometric information, rather than object-level knowledge needed for self-supervision.

Our method intends to exploit the properties of both categories. As large outdoor scenes are composed of multiple objects and surfaces, we propose to learn the occupancy using a local approach, but with the objective of learning object-level representations.

Supervision. Thanks to the synthetic datasets such as ModelNet [84], ShapeNet [13] or ABC [41], it is possible to obtain an occupancy ground truth used for supervision. Local approaches supervised on these shapes are able to generalize to scene level when provided an additional orientation information such as point normals [6, 40] or sensor location [74].

Other methods have been developed to perform volume reconstruction without the need for any ground truth label, focusing on loss design. Sign agnostic losses are used in SAL [2] and SALD [3]. It minimizes the norm of the estimated signed distance field with respect to input-computed distance field. A careful initialization of the network is needed to ensure a signed output. IGR [33] encourages a null distance at each input point, while enforcing a non-null gradient norm, thus favoring sign changes. Needrop [7] uses a loss applied on a segment sampled such that the middle point is an input point and force both extremities to have opposite labels.

Our approach, in contrast, does not rely on the design of a specific loss. We use the sensor information as in [74] to generate points where we can estimate the occupancy with confidence. We can then train our model with a binary cross-entropy as if in a supervised setting.

3. Method

We propose surface reconstruction as a pretext task to create self-supervised features for 3D point clouds, that are well suited for semantic segmentation and object detection.

As mentioned in Section 2, networks can be trained with supervision to estimate implicitly the surface underlying a given point cloud. Besides, using visibility information has been shown an effective way to improve surface reconstruction by adding extra points to supervise the training [74]. Inspired by these works, we propose to reuse rich shape features for downstream tasks. To create such features without the need for manual annotations, we propose to use visibility information for unsupervised surface reconstruction. What's more, we adapt surface reconstruction to produce intermediate latent vectors that capture not only geometrical details but also some semantic knowledge. The overall principle of the method is presented in Figure 2.

3.1. Support points and latent vectors

In methods such as POCO [6], resp. ConvONet [68], a latent vector is first computed for each input point, resp. each pre-defined (2D or 3D) grid point. The occupancy of a given query point is then predicted from the latent vectors of neighboring points [6], resp. neighboring grid points [68].

Similarly, in our setting, we consider that input points $p \in P$, with optional intensity i_p, are given to a backbone that outputs, on given support points $s \in S$, an associated latent vector z_s. For semantic segmentation, the support points are the input points. For object detection, using detectors such as SECOND [87] or PVRCNN [73], the support points are 2D points on a grid in the bird-eye-view (BEV) plane.

3.2. Generating query points for self-supervision

To introduce self-supervision, we create query points $q \in Q$ with known occupancy o_q, although no ground truth surface is used or even available. To that end, we exploit visibility information, knowing the sensor location.

Given a 3D point p sampled on the surface of the scene by a sensor whose center is located at c, we consider that points *in front of* p, i.e., on the 3D segment $[c, p]$, are empty, while points immediately *behind* the observed point p along the line of sight of the sensor are not (see Figure 3(a)).

Concretely, as in [74], for each input point p, we create two query points $q_{\text{front}} = p - \delta_p$ and $q_{\text{behind}} = p + \delta_p$, where $\delta_p = \delta u$, $\delta > 0$ is a small distance, and $u = (c-p)/\|c-p\|$ is a unit vector pointing from the sensor location c to the observed point p. q_{front} is considered empty and q_{behind} full. While q_{front} is empty for sure (unless p is an outlier), q_{behind} is not necessarily full in case the object is very thin (thinner than δ) or at the border of objects for grazing lines of sight. Nevertheless, [74] shows that this hypothesis is correct enough in general, leading to significant benefits in surface reconstruction. The parameter δ however has to be adjusted to the expected minimum thickness of scene objects.

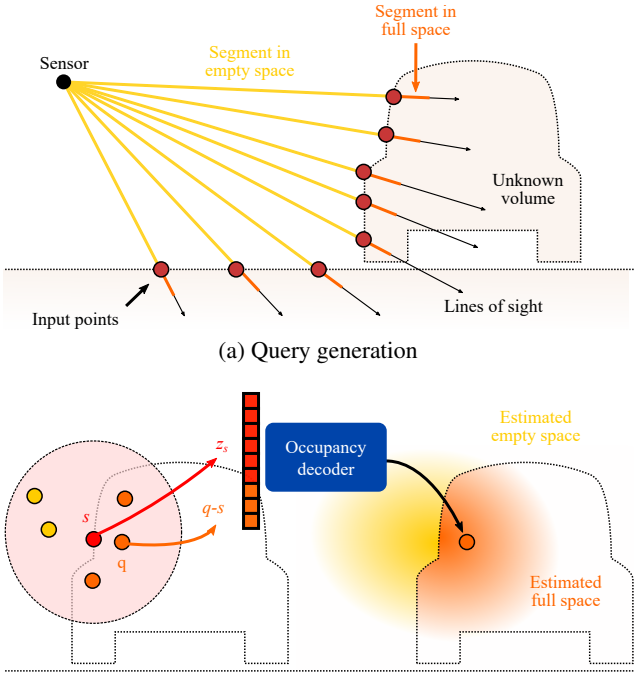


Figure 3. Generation of the queries. The line between the sensor and each points is used to find empty and full points in space. Input features for each query are constructed with the features of support points and the relative coordinates of the query to those points.

Additionally, we create a third empty query point q_{sight} randomly picked in the segment $[c, p]$. These query points $q \in Q$ and associated occupancies o_q are used as the supervision signal to pre-train the backbone.

3.3. Estimating occupancy towards semantization

In surface reconstruction methods, the goal is to infer the most accurate reconstruction. To that end, these methods rely on different forms of interpolation [6, 68], gathering information from latent vectors of neighboring support points.

Here, our goal is different. We do not care so much about geometrical details. What we want is to infer features that are typical to the underlying objects or object parts. To that end, we reverse the reconstruction paradigm: instead of estimating occupancy by combining fine local information from neighbors, we encourage features of neighbors to be similar by enforcing the feature of a point to be able to reconstruct a whole ball around it. Surface reconstruction is then possibly less accurate, but the inferred features are more global to the object or object part, which makes semantization easier.

Concretely, for each support point s, we consider as neighborhood a ball of radius r centered at s, as well as the queries $Q_s = \{q \in Q, \|q - s\| \leq r\}$ falling in this neighborhood.

For each such query $q \in Q_s$, we create an input to the occupancy decoder which is the concatenation of the latent vector z_s and the local coordinates of the query with respect to the support, i.e., $q - s$. The occupancy decoder consists of an MLP and a final sigmoid activation function. For each input vector $z_s \oplus (q - s)$, it produces the estimated occupancy at q given the latent vector at s, denoted $\hat{o}_{q|s}$. This predicted occupancy has value in $[0, 1]$; it corresponds to an empty (resp. full) space if greater (resp. less) than 0.5.

3.4. Reconstruction loss

Intuitively, we want the loss function to encourage the latent vector z_s of a support point s to be able to reconstruct everything inside the ball of radius r centered at s. To do so practically, the loss function $\mathcal{L}_{\mathsf{occup}}$ penalizes wrong estimated occupancies for query points q falling into the ball. Concretely, $\mathcal{L}_{\mathsf{occup}}$ is a binary cross-entropy between the estimated occupancies at the query points $\hat{o}_{q|s}$ and the actual sensor-based self-supervised occupancies o_q:

$$\mathcal{L}_{\mathsf{occup}} = \frac{-1}{|S|} \sum_{s \in S} \frac{1}{|Q_s|} \sum_{q \in Q_s} o_q \log(\hat{o}_{q|s}) + (1 - o_q) \log(1 - \hat{o}_{q|s}). \tag{1}$$

where $|S|$ is the number of support points and $|Q_s|$ is the number of query points in the ball centered on S. Please note that a query point may appear several times in this term, as it may be in the neighborhood of several support points.

3.5. Particular case of BEV support points

Current object detectors, such as SECOND [87] or PV-RCNN [73], operate a projection on the bird-eye-view (BEV) plane, which has no pre-defined altitude. In this case, instead of considering a ball centered on s as query neighborhood Q_s for support point s, we define Q_s as the infinite vertical cylinder of radius r centered on s. As previously, a latent vector z_s should be able to estimate the occupancy of all query points falling in the cylinder.

Please note that the 3D coordinates of q are still used to compute the input to the occupancy decoder, using a dummy $Z = 0$ vertical coordinate to compute the relative location $q - s$ that is provided as input to the decoder.

3.6. Exploiting returned lidar intensity

Our approach is based on volumetric occupancy estimation given an input point cloud, which is a purely geometric task. However, lidar point clouds often also come with returned lidar intensity at each point, which is widely used to enrich the input features for both semantic segmentation and object detection [44, 50, 54, 73, 87, 94]. We follow the literature and also input the lidar intensity when available.

Nevertheless, it is not obvious that the network makes the most of both geometry and intensity information. We thus consider here an extra intensity-based loss term, which our ablation study proves beneficial. In this case, we consider a variant of the above presentation where the decoder outputs not only the estimated occupancy $\hat{o}_{q|s}$ but also an estimated intensity $\hat{i}_{q|s}$ of the input point p used to generate query q. We then introduce an intensity-recovery loss term $\mathcal{L}_{\mathsf{intens}}$ that penalizes the ℓ_2 distance between the estimated intensity $\hat{i}_{q|s}$ of the query point q and the actual intensity $i_q = i_p$ of the corresponding input point p:

$$\mathcal{L}_{\mathsf{intens}} = \frac{1}{|S|} \sum_{s \in S} \frac{1}{|Q'_s|} \sum_{q \in Q'_s} ||\hat{i}_{q|s} - i_q||_2 \tag{2}$$

where $Q'_s \subset Q_s$ is the subset of query points $q \in Q_s$ consisting only of queries q_{front} and q_{behind} close to sampled points, ignoring queries q_{sight} lying between the sensor and the points, as intensity does not make sense for them. In this setting, the overall loss function is defined as:

$$\mathcal{L} = \mathcal{L}_{\mathsf{occup}} + \lambda \mathcal{L}_{\mathsf{intens}} \tag{3}$$

where λ is a weight for balancing the two terms.

4. Experiments

To assess our pre-training method, we conduct experiments on both semantic segmentation and object detection.

4.1. Datasets

We briefly describe the datasets used for pre-training semantic segmentation (Pre-Seg), downstream semantic segmentation (Seg), pre-training detection (Pre-Det), downstream object detection (Det).

nuScenes [8] (Pre-Seg, Seg, Pre-Det) is composed of 1000 sequences (train/val/test) acquired with a 32-layer rotative lidar in Boston and Singapore. Points are annotated with 15 classes. Ablation sets and partial training sets are defined according to [72] (for 0.1% we use sequence 0392).

SemanticKITTI [5] (Pre-Seg, Seg) contains 22 lidar sequences acquired with a 64-layers Velodyne HDL-64E sensor annotated with 19 labels. For downstream task, partial training sets are those defined in [64].

SemanticPOSS [65] (Seg) is composed of 6 sequences annotated with the same labels as SemanticKITTI. Contrary to [64], we use the official validation set (sequence 3).

LivoxSimu [51] (Pre-Seg, Seg) is a synthetic dataset simulating 5 Livox Horizon lidars. Points are annotated with 14 labels. The first 90% of the data is used as training set, the remaining is used as the test set.

ONCE (Pre-Det, Det) contains 1M lidar scenes, most of which are unannotated. Pre-training is done with the small unlabeled set as in [54], while downstream uses the training and validation splits to train and evaluate the detectors.

KITTI 3D [27] (Pre-Det, Det) is a dataset dedicated to various autonomous driving tasks, including 3D detection.

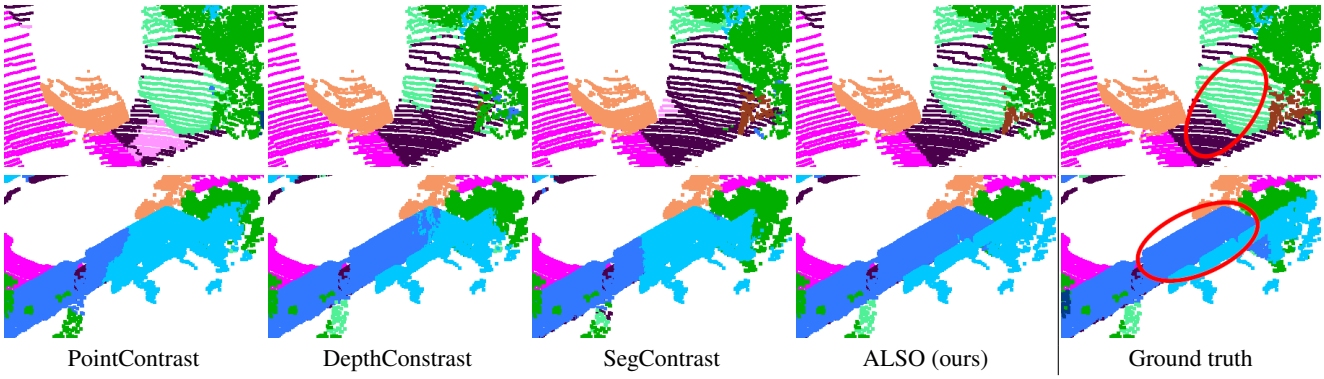


Figure 4. Qualitative comparison on SemanticKITTI segmentation, fine-tuned with 1% of training data.

Annotations are provided for ∼7.5k frames, in the front camera field of view. We provide evaluation scores on the moderate difficulty objects with the official 40 recall points R_{40}, and 11 recall points R_{11} for comparison purpose.

KITTI-360 (Pre-Det) is a multiple sensor dataset, including 100k lidar scans in a suburban environment. We use this dataset for detection pre-training purpose.

4.2. Semantic segmentation

Network architectures. To evaluate the ability of our approach to generalize to different architectures and for a fair comparison with previous work, we use several backbones. Experiments are done with: two variants of MinkUNet [21], one from [72, 86], one from [64], and SPVCNN [76]. The occupancy decoder is a 4 layer-MLP with interposed ReLU activations and a hidden size of 128, similar to the latent output size. For segmentation fine-tuning, the occupancy head is removed, as well as the last layer of the backbone, which is replaced by a linear layer with output size corresponding to the number of classes in the dataset.

Pre-training. We use the AdamW optimizer with default PyTorch parameters: learning rate 10^{-3}, $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 10^{-8}$ and weight decay 0.01. For nuScenes and LivoxSimu (resp. SemanticKITTI), we downsample the input to 16k points (resp. 80k), randomly select 2k query points (resp. 4k) per frame and pre-train for 200 epochs with batch size 16 (resp. 50 epochs, with batch size 4).

We set $\delta = 10$ cm. It corresponds more or less to the minimal thickness of objects encountered in outdoor scenes, e.g. poles or human limbs. We use $\lambda = 1$ in all our experiments.

Downstream training. Each method's pre-training is evaluated after fine-tuning on downstream tasks.

Outdoor data. For fair comparison and in order to setup a simple evaluation protocol, we use the cross-entropy loss and AdamW default PyTorch parameters for all the downstream experiments: learning rate 10^{-3}, $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 10^{-8}$ and weight decay 0.01. In addition, the learning rate is modulated by a cosine annealing scheduler [52] with multiplier ranging from 1 at first epoch to 0. The fluctuation for any given metric during different runs of the same experiment being significant, we report averages over 5 runs, both for our method, baselines and concurrent works, whenever possible. For nuScenes and LivoxSimu (resp. for SemanticKITTI and SemanticPOSS), we use 16k (resp. 80k) input points, batch size 8 (resp. 2). For final score computation, we reproject the labels of the downsampled point cloud on the original point cloud with nearest-neighbor interpolation. We adapt the number of epochs according to the percentage of training data used. Note that we use the same number of epochs for all datasets: 1000 for 0.1%; 500 for 1%; 100 for 10%; 50 for 50% and 30 for 100%.

Ablation studies. Ablation studies are done on nuScenes. We follow the evaluation procedure of [72] where the training set of nuScenes is split in ablation-train and ablation-val sets, and fine-tune using 1% of the annotations of the ablation-train set. Doing so ensures that we do not tune method parameters on the validation set which is the set used for comparison to other methods. We train for 100 epochs. Ablations are presented in Table 3.

Context radius. We study the context radius r to be used for outdoor lidar dataset (Table 3(a)). r is a crucial parameter. On the one hand, a small value makes it easier for the network to reconstruct the occupancy, however it will only contain local geometric features without object level shape understanding. On the other hand, a too large value may cover an area that includes several objects/surfaces, which thus will not favor discrimination between classes/objects in the latent vector. Intuitively, r should correspond to the scale of the objects that we want to discover in the scene.

Intensity loss. Next, we study the impact of intensity both as an input and as a reconstruction objective (Table 3(b)). It appears that providing intensity to the network already increases the performances (second column). An additional boost is obtained when using $\mathcal{L}_{\text{intens}}$ (third column), enforcing the network to maintain intensity information, which then can easily be used at downstream time.

Method evaluation. In this section, we evaluate our method on several datasets and for several sensor types.

Dataset	Backbone	Method	0.1%		1%		10%		50%		100%	
nuScenes	MinkUNet [72]	No pre-training	21.6		35.0		57.3		69.0		71.2	
		PointContrast [86]	27.1	+5.5	37.0	+2.0	58.9	+1.6	69.4	+0.4	71.1	-0.1
		DepthContrast [93]	21.7	+0.1	34.6	-0.4	57.4	+0.1	69.2	+0.2	71.2	-
		ALSO (ours)	26.2	+4.6	37.4	+2.4	59.0	+1.7	69.8	+0.8	71.8	+0.6
	SPVCNN [76]	No pre-training	22.2		34.4		57.1		69.0		70.7	
		ALSO (ours)	24.8	+2.6	37.4	+3.0	58.4	+1.3	69.5	+0.5	71.3	+0.6
SemanticKITTI	MinkUNet [64]	No pre-training	30.0		46.2		57.6		61.8		62.7	
		PointContrast [86]	32.4	+2.4	47.9	+1.7	59.7	+2.1	62.7	+0.9	63.4	+0.7
		DepthContrast [93]	32.5	+2.5	49.0	+2.8	60.3	+2.7	62.9	+1.1	63.9	+1.2
		SegContrast [64]	32.3	+2.3	48.9	+2.7	58.7	+1.1	62.1	+0.3	62.3	-0.4
		ALSO (ours)	35.0	+5.0	50.0	+3.8	60.5	+2.9	63.4	+1.6	63.6	+0.9
	SPVCNN [76]	No pre-training	30.7		46.6		58.9		61.8		62.7	
		ALSO (ours)	35.0	+4.3	49.1	+2.5	60.6	+1.7	63.6	+1.8	63.8	+1.1
SemanticPOSS (pre-training SemanticKITTI)	MinkUNet [64]	No pre-training	36.9		46.4		54.5		55.3		55.1	
		PointContrast [86]	39.3	2.4	48.1	1.7	55.1	+0.6	56.2	+0.9	56.2	+1.1
		DepthContrast [93]	39.7	+2.8	48.5	+2.1	55.8	+1.3	56.0	+0.7	56.5	+1.4
		SegContrast [64]	41.7	+4.8	49.4	+3.0	55.4	+0.9	56.2	+0.9	56.4	+1.3
		ALSO (ours)	40.7	+3.8	49.6	+3.2	55.8	+1.3	56.4	+1.1	56.7	+1.6
LivoxSimu	MinkUNet [72]	No pre-training	48.0		63.8		66.7		68.5		68.9	
		ALSO (ours)	52.6	+4.6	65.5	+1.7	67.8	+1.1	69.6	+1.1	69.7	+0.8

Results averaged over 5 runs. Individual run details and standard deviation in the supplementary material.

Table 1. Semantic segmentation results. We report the mIoU (%) of fine-tuned models on four different datasets, while varying the amount of annotated data, the pre-training dataset and the architecture. We compare ALSO against a non-pre-trained baseline and concurrent work.

Quantitive scores are presented in Table 1.

First, we compare ALSO to state-of-the-art methods PointContrast [86], DepthContrast [93] and SegContrast [64] on nuScenes, SemanticKITTI and SemanticPOSS. Compared to [64], using the default AdamW parameters and training for longer leads to improved performances for all models, including the "no pre-training" setting. We can observe that performance ranking among the previously cited methods may vary from one dataset to the other, more particularly, DepthContrast benefits a lot from a higher point cloud density. ALSO outperforms the previous methods for nearly all the configurations (datasets and percentages), e.g., ranking first on nuScenes 1% by 0.4 mIoU point over PointContrast and first on SemanticKITTI 0.1% by 2.5 points over DepthContrast. Qualitative examples are presented in Figure 4. We demonstrate here that occupancy reconstruction pre-training method is an efficient and valid alternative to the memory costly contrastive methods.

Second, to hightlight that ALSO is not architecture dependant, we also experiment with SPVCNN [76], a sparse variant of PVCNN [50] mixing local point based representation and sparse voxel convolutions. Our approach presents the same improvement margin over from scratch training than with a MinkUNet.

Third, nuScenes, SemanticKITTI and SemanticPOSS are all datasets created using a rotative lidar. Even though they were acquired with different sensors, they present similar patterns on the surface, i.e., concentric circles. We test our approach on the LivoxSimu dataset, where sensors have a very different acquisition patterns. The MinkUNet network shows a similar behavior as for the rotative sensors, highlighting that our approach can work with different sensors.

4.3. Detection

Network architectures. For detection, we experiment with the commonly used SECOND [87] and PV-RCNN [76] object detectors. They share the same backbone architecture: a 3D sparse encoder (3D-backbone) made with 3D sparse convolution processing input voxels, and a bird-eye-view (BEV) encoder (2D-backbone) applied after BEV projection. They mainly differ by the detection heads: SECOND directly applies a region prosal network (RPN) on top of the 2D-backbone, PV-RCNN uses a point-level refinement of the RPN predictions, leading to more accurate boxes and confidence estimation. We use the OpenPCDet [77] implementation of these networks.

Pre-training. We pre-train the detection backbone (3D and 2D) using ALSO. As for semantic segmentation, we train with the default AdamW optimizer parameters, limiting the number of input points to 80k (to prevent high

(a) KITTI3D [27], validation set, moderate difficulty.

Method	Data.	Cars	Ped.	Cycl.	mAP	Diff.
SECOND - R_{40} metric						
No pre-training	-	81.50	48.82	65.72	65.35	
ALSO (ours)	K	81.97	51.93	69.14	67.68	+2.33
	K360	81.79	52.45	70.68	68.31	+2.96
	NS	81.78	54.24	68.19	68.07	+2.72
SECOND - R_{11} metric						
No pre-training	-	78.62	52.98	67.15	66.25	
Voxel-MAE [58]	K	78.90	53.14	68.08	66.71	+0.46
ALSO (ours)	K	78.78	53.57	68.22	66.86	+0.61
	K360	78.63	54.23	69.35	67.40	+1.15
	NS	78.65	55.17	68.05	67.29	+1.04
PV-RCNN - R_{40} metric						
No pre-training	-	84.50	57.06	70.14	70.57	
STRL [39]	K	84.70	57.80	71.88	71.46	+0.89
GCC-3D [46]	NS	-	-	-	70.75	+0.18
GCC-3D [46]	W	-	-	-	71.26	+0.69
PointCont. [86]	W	84.18	57.74	72.72	71.55	+0.98
Prop.Cont. [89]	W	84.72	60.36	73.69	72.92	+2.35
ALSO (ours)	K	84.72	58.49	75.06	72.76	+2.19
	K360	84.68	60.16	74.04	72.96	+2.39
	NS	84.86	57.76	74.98	72.53	+1.98
PV-RCNN - R_{11} metric						
No pre-training	-	83.61	57.90	70.47	70.66	
Voxel-MAE [58]	K	83.82	59.37	71.99	71.73	+1.07
ALSO (ours)	K	83.67	58.48	73.74	71.96	+1.30
	K360	83.39	60.83	73.85	72.69	+2.03
	NS	83.77	58.49	74.35	72.20	+1.54

(b) ONCE [54], validation set, SECOND detector, ONCE metric.

Method	Data.	Cars	Ped.	Cycl.	mAP	Diff.
No pre-training	-	71.19	26.44	58.04	51.89	
BYOL [32]	O_s	68.02	19.50	50.61	46.04	-5.85
PointCont. [86]	O_s	71.07	22.52	56.36	49.98	-1.91
SwAV [11]	O_s	72.71	25.13	58.05	51.96	+0.07
DeepCluster [10]	O_s	73.19	24.00	58.99	52.06	+0.17
ALSO (ours)	O_s	71.73	28.16	58.13	52.68	+0.79

Datasets: KITTI3D (K), KITTI-360 (K360), nuScenes (NS), ONCE Small (O_s), Waymo (W).

Table 2. Detection results on KITTI3D (a) and ONCE (b). We report AP (%) and the dataset used for pre-training each method.

memory peeks), and query points to 4k. For KITTI3D, we pre-train with batch size 8 for 500 epochs on KITTI3D, 100 on nuScenes and 75 on KITTI-360. For ONCE, we pre-train for 50 epochs on the U_{small} unannotated set.

Downstream training. Downstream is done with OpenPCDet [77] for KITTI3D and ONCE [54] official detection code, in both cases with default settings.

Quantitative evaluation. Table 2 displays the scores obtained using our pre-training pipeline.

Pre-training on the downstream dataset. First, we look specifically at methods trained on the target dataset KITTI3D (K) in Table 2(a) and on ONCE, Table 2(b). We can observe that we consistently improve over the no-pre-training baseline: +2% with SECOND, +2.2% with PV-RCNN on KITTI (R_{40}), and by +0.8% on ONCE. Compared to literature, we perform on par with Voxel-MAE [58].

Transfering from another dataset. Finally, we also pre-trained on KITTI360 and nuScenes to assess the transferability of our pre-training. We observe first that pre-training on larger datasets, leads to higher performances, regardless of the dataset, even if the sensor is not the same. Then, when considering the choice of the pre-training dataset as a design option, we reach the state of the art on par with ProposalContrast [89] trained on the Waymo dataset [75].

(a) Reconstruction radius (with intensity and $\mathcal{L}_{\text{intens}}$)

Radius (m)	0.5	1	2	4
mIoU (%)	37.6	**38.4**	38.2	36.4

(b) Intensity for pre-training (with radius=1.0 m)

Input intensity	✗	✓	✓
Loss $\mathcal{L}_{\text{intens}}$	✗	✗	✓
mIoU (%)	36.4	38.2	**38.4**

Table 3. Parameter study for radius r (a) and ablation study of intensity usage (b), as input and objective for pre-training. Evaluation is on the ablation-val set of nuScenes, training 100 epochs.

5. Conclusion

In this work we investigate the use of occupancy reconstruction as a pretext task for self-supervision on point cloud. We show that a supervision signal for occupancy estimation can directly be inferred from the sensor information. The resulting method is conceptually simple and can be trained with limited resources (single 16GB memory GPU). ALSO can be used for semantic segmentation as well as for detection, and provides clear benefits on tested architectures and datasets. It outperforms contrastive methods on semantic segmentation and is able to perform on par with state-of-the-art detection methods specifically designed for the task.

In the footsteps of input reconstruction approaches, we show that geometric tasks, here estimating the occupancy, are meaningful alternatives to contrastive learning and masked autoencoders. Future work include studying combinations with contrast-based and completion-based approaches.

Acknowledgements: This work was supported in part by the French Agence Nationale de la Recherche (ANR) grant MultiTrans (ANR21-CE23-0032). This work was performed using HPC resources from GENCI–IDRIS (Grants 2021-AD011012883 and 2022-AD011012883R1).

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3D point clouds. In *ICML*, 2018. 3

[2] Matan Atzmon and Yaron Lipman. SAL: Sign agnostic learning of shapes from raw data. In *CVPR*, 2020. 3

[3] Matan Atzmon and Yaron Lipman. SALD: Sign agnostic learning with derivatives. In *ICLR*, 2021. 3

[4] Hangbo Bao, Li Dong, and Furu Wei. Beit: Bert pre-training of image transformers. In *ICLR*, 2021. 2

[5] 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. 5

[6] Alexandre Boulch and Renaud Marlet. POCO: Point convolution for surface reconstruction. In *CVPR*, pages 6302–6314, 2022. 3, 4

[7] Alexandre Boulch, Gilles Puy, and Renaud Marlet. NeeDrop: Self-supervised shape representation from sparse point clouds using needle dropping. In *3DV*, 2021. 3

[8] 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. 5

[9] Ruojin Cai, Guandao Yang, Hadar Averbuch-Elor, Zekun Hao, Serge Belongie, Noah Snavely, and Bharath Hariharan. Learning gradient fields for shape generation. In *ECCV*, 2020. 3

[10] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and M. Douze. Deep clustering for unsupervised learning of visual features. In *ECCV*, pages 132–149, 2018. 2, 8

[11] 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, 8

[12] 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. 2

[13] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. ShapeNet: An information-rich 3D model repository. *arXiv:1512.03012*, 2015. 3

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

[15] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020. 2

[16] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv:2003.04297*, 2020. 2

[17] 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*, 2021. 3

[18] Z. Chen and H. Zhang. Learning implicit fields for generative shape modeling. In *CVPR*, 2019. 3

[19] J. Chibane, T. Alldieck, and G. Pons-Moll. Implicit functions in feature space for 3D shape reconstruction and completion. In *CVPR*, 2020. 3

[20] Julian Chibane, Aymen Mir, and Gerard Pons-Moll. Neural unsigned distance fields for implicit function learning. In *NeurIPS*, 2020. 3

[21] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4D spatio-temporal ConvNets: Minkowski convolutional neural networks. In *CVPR*, 2019. 1, 6

[22] Carl Doersch, Abhinav Gupta, and Alexei Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015. 2

[23] 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*, 2021. 3

[24] Philipp Erler, Paul Guerrero, Stefan Ohrhallinger, Niloy J. Mitra, and Michael Wimmer. Points2Surf: Learning implicit surfaces from point clouds. In *ECCV*, 2020. 3

[25] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, volume 96-34, pages 226–231, 1996. 3

[26] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *CACM*, 24(6):381–395, 1981. 3

[27] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 5, 8

[28] 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. 2

[29] 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

[30] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018. 2

[31] G. Gkioxari, J. Malik, and J. Johnson. Mesh R-CNN. In *ICCV*, 2019. 3

[32] 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, 8

[33] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. In *ICML*, 2020. 3

[34] T. Groueix, M. Fisher, V.G. Kim, B.C. Russell, and M. Aubry. AtlasNet: A papier-mâché approach to learning 3D surface generation. In *CVPR*, 2018. 3

[35] Kaveh Hassani and Mike Haley. Unsupervised multi-task feature learning on point clouds. In *ICCV*, 2019. 3

[36] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022. 2

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

[38] Peiyun Hu, Jason Ziglar, David Held, and Deva Ramanan. What you see is what you get: Exploiting visibility for 3D object detection. In *CVPR*, 2020. 2

[39] Siyuan Huang, Yichen Xie, Song-Chun Zhu, and Yixin Zhu. Spatio-temporal self-supervised representation learning for 3D point clouds. In *ICCV*, 2021. 2, 3, 8

[40] C. Jiang, A. Sud, A. Makadia, J. Huang, M. Nießner, and T. Funkhouser. Local implicit grid representations for 3D scenes. In *CVPR*, 2020. 3

[41] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. ABC: A big cad model dataset for geometric deep learning. In *CVPR*, 2019. 3

[42] Abhijit Kundu, Yin Li, and James M Rehg. 3D-RCNN: Instance-level 3D Object Reconstruction via Render-and-Compare. In *CVPR*, 2018. 2

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

[44] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019. 1, 5

[45] 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

[46] 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. 3, 8

[47] Y. Liao, S. Donne, and A. Geiger. Deep marching cubes: Learning explicit surface representations. In *CVPR*, 2018. 3

[48] C. Lin, C. Kong, and S. Lucey. Learning efficient point cloud generation for dense 3D object reconstruction. In *AAAI*, 2018. 3

[49] 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

[50] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel CNN for efficient 3D deep learning. In *NeurIPS*, 2019. 5, 7

[51] Livox. Livox Simu dataset. https://www.livoxtech.com/simu-dataset. 5

[52] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *ICLR*, 2017. 6

[53] Yiming Luo, Zhenxing Mi, and Wenbing Tao. DeepDT: Learning geometry from Delaunay triangulation for surface reconstruction. In *AAAI*, 2021. 3

[54] 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. 2, 5, 8

[55] D. Maturana and S. Scherer. VoxNet: A 3D convolutional neural network for real-time object recognition. In *IROS*, 2015. 3

[56] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy networks: Learning 3D reconstruction in function space. In *CVPR*, 2019. 3

[57] M. Michalkiewicz, J.K. Pontes, D. Jack, M. Baktashmotlagh, and A. Eriksson. Implicit surface representations as layers in neural networks. In *ICCV*, 2019. 3

[58] Chen Min, Dawei Zhao, Liang Xiao, Yiming Nie, and Bin Dai. Voxel-MAE: Masked autoencoders for pre-training large-scale point clouds. In *WACV*, 2023. 2, 3, 8

[59] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *CVPR*, 2020. 2

[60] Ishan Misra, Lawrence Zitnick, and Martial Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *ECCV*, 2016. 2

[61] Mahyar Najibi, Guangda Lai, Abhijit Kundu, Zhichao Lu, Vivek Rathod, Thomas Funkhouser, Caroline Pantofaru, David Ross, Larry S Davis, and Alireza Fathi. DOPS: Learning to detect 3D objects and predict their 3D shapes. In *CVPR*, 2020. 2

[62] Charlie Nash, Yaroslav Ganin, S. M. Ali Eslami, and Peter W. Battaglia. PolyGen: an autoregressive generative model of 3D meshes. In *ICML*, 2020. 3

[63] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016. 2

[64] 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. 2, 3, 5, 6, 7

[65] Yancheng Pan, Biao Gao, Jilin Mei, Sibo Geng, Chengkun Li, and Huijing Zhao. SemanticPOSS: A point cloud dataset with large quantity of dynamic instances. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 687–693. IEEE, 2020. 5

[66] 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

[67] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019. 3

[68] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *ECCV*, 2020. 3, 4

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

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

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

[72] 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. 2, 3, 5, 6, 7

[73] 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*, pages 10526–10535, 2020. 1, 3, 4, 5

[74] Raphael Sulzer, Loic Landrieu, Alexandre Boulch, Renaud Marlet, and Bruno Vallet. Deep surface reconstruction from point clouds with visibility information. In *ICPR*, 2022. 3, 4

[75] 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*, 2020. 8

[76] Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3D architectures with sparse point-voxel convolution. In *ECCV*, 2020. 1, 6, 7

[77] OpenPCDet Development Team. OpenPCDet: An open-source toolbox for 3D object detection from point clouds. https://github.com/open-mmlab/OpenPCDet, 2020. 7, 8

[78] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. In *ICLR*, 2020. 2

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

[80] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.G. Jiang. Pixel2Mesh: Generating 3D mesh models from single RGB images. In *ECCV*, 2018. 3

[81] 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*, 2021. 3

[82] Francis Williams, Zan Gojcic, Sameh Khamis, Denis Zorin, Joan Bruna, Sanja Fidler, and Or Litany. Neural fields as learnable kernels for 3D reconstruction. In *CVPR*, 2022. 3

[83] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *CVPR*, 2015. 3

[84] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *CVPR*, 2015. 3

[85] Zhirong Wu, Yuanjun Xiong, Stella Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance-level discrimination. In *CVPR*, 2018. 2

[86] 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. 2, 3, 6, 7, 8

[87] Yan Yan, Yuxing Mao, and Bo Li. SECOND: Sparsely embedded convolutional detection. *Sensors*, 18, 2018. 1, 3, 4, 5, 7

[88] G. Yang, X. Huang, Z. Hao, M. Liu, S.J. Belongie, and B. Hariharan. PointFlow: 3D point cloud generation with continuous normalizing flows. In *ICCV*, 2019. 3

[89] 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, 3, 8

[90] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3D object detection and tracking. In *CVPR*, 2021. 1

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

[92] Richard Zhang, Phillip Isola, and Alexei Efros. Colorful image colorization. In *ECCV*, 2016. 2

[93] Zaiwei Zhang, Rohit Girdhar, Armand Joulin, and Ishan Misra. Self-supervised pretraining of 3D features on any point-cloud. In *ICCV*, 2021. 2, 3, 7

[94] 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*, 2021. 1, 5