Image-to-Lidar Self-Supervised Distillation for Autonomous Driving Data –Supplementary material–

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

A. Complementary Results

A.1. Visual Inspection

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

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

Backbone $g_{\bar{\omega}_{\text{bck}}}$	Pretrain	mIoU
ResNet-50	Full sup.	39.2
ResNet-50	MoCov2 [4]	39.2
ViT-S/16	DINO [3]	39.5

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

We also provide in the supplementary material a video illustrating the capacity of our pre-trained model in doing semantic segmentation on a sequence of point clouds, without fine-tuning. The video is generated as follows. We choose a scene from the validation set of nuScenes [2]. We collect all the point features along with the class labels in the first frame of the sequence. This set constitutes our annotated database. We consider three classes: 'vegetation', 'car' and 'pedestrian'. The points in the following frames are classified by using a binary k-NN classifier (k=20) for each of these class. We display the predicted probability for each class in all subsequent frames: red for 'car', blue for 'pedestrian', green for 'vegetation'.[1] We notice that we are able to classify correctly several points in of each of the considered classes throughout the whole sequence. In particular, we detect correctly pedestrians at the beginning of sequence and cars at the end of the sequence. Some points are misclassified but we recall these results are obtained without any fine-tuning of the backbone.

A.2. Choice of the Image Backbone

We present in Tab. 5 the performance reached with SLidR on nuScenes semantic segmentation when using a ResNet-50 pre-trained under full supervision on ImageNet

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

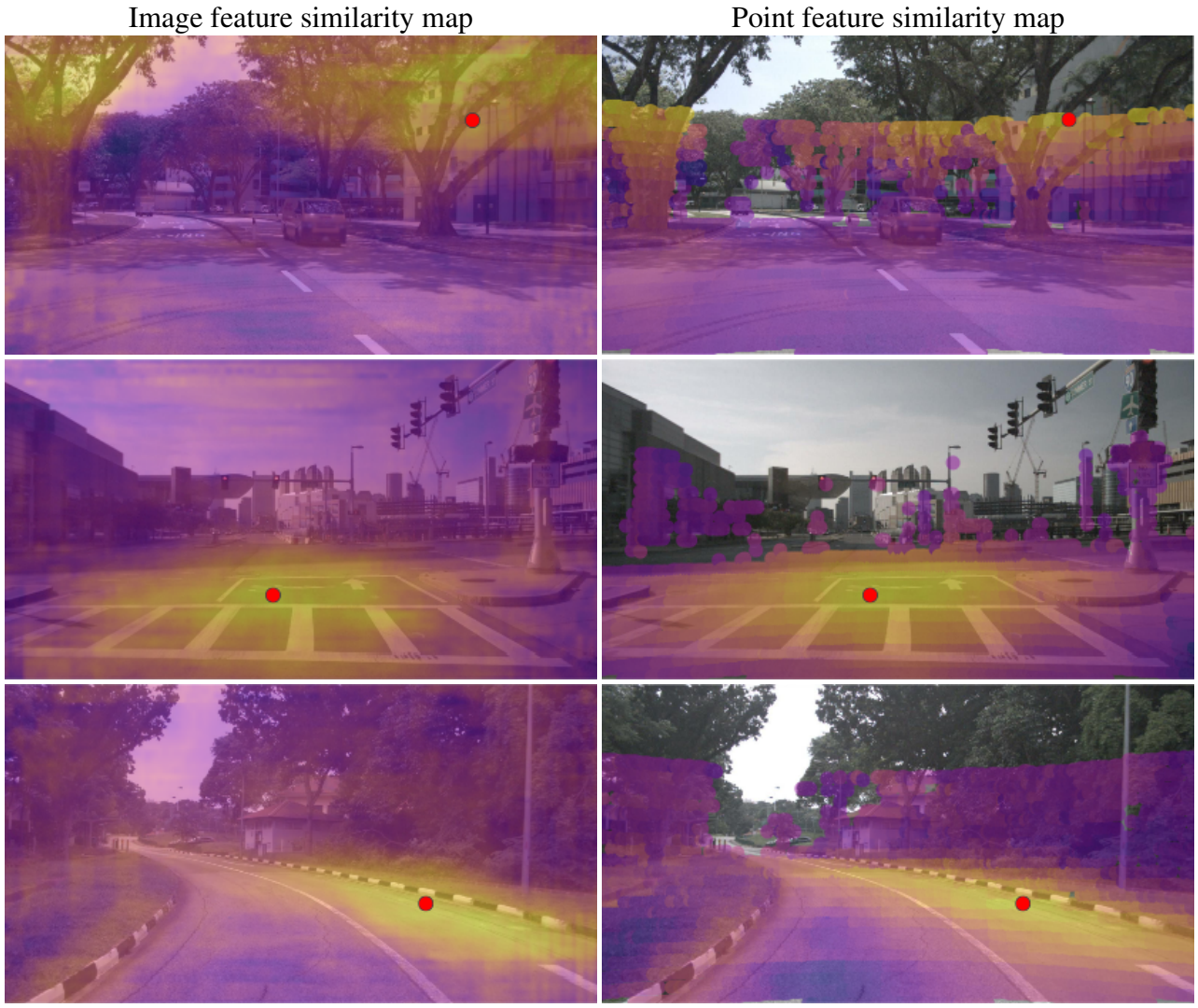

Figure 4. Cosine similarity between the SLidR's feature of a query point (displayed as a red dot) and: (a) the pixel features of an image in the same scene (left column - image feature similarity map); (b) the features of the other points projected in the same image (right column - point feature similarity map). The colormap goes from violet to yellow for respectively low and high similarity scores. We show these maps for two scenes in the validation set of [2].

or under self-supervision using MoCov2. We notice that there is no loss of performance because of the use of self-supervision to pretrain $g_{\bar{\omega}_{\text{bck}}}$.

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

A.3. Choice of the Superpixels Method

We present in Tab. 6 the impact of the choice of the number of superpixels or of the superpixels algorithm on the performance of SLidR. For Felzenszwalb's [5] method (called FH), we used a scale parameter of 300, a gaussian pre-processing of standard deviation 0.35 and a superpixels minimal size of 4000 pixels, which yield at most 143 superpixels per image on nuScenes' training set. For SLIC, we tested 100, 150 or 200 superpixels per image. We see that it is important to adjust the number of superpixels correctly to avoid too much over-segmentation and under-segmentation which both impact negatively the performance. The results in the main paper are obtained with SLIC and 150 superpix-

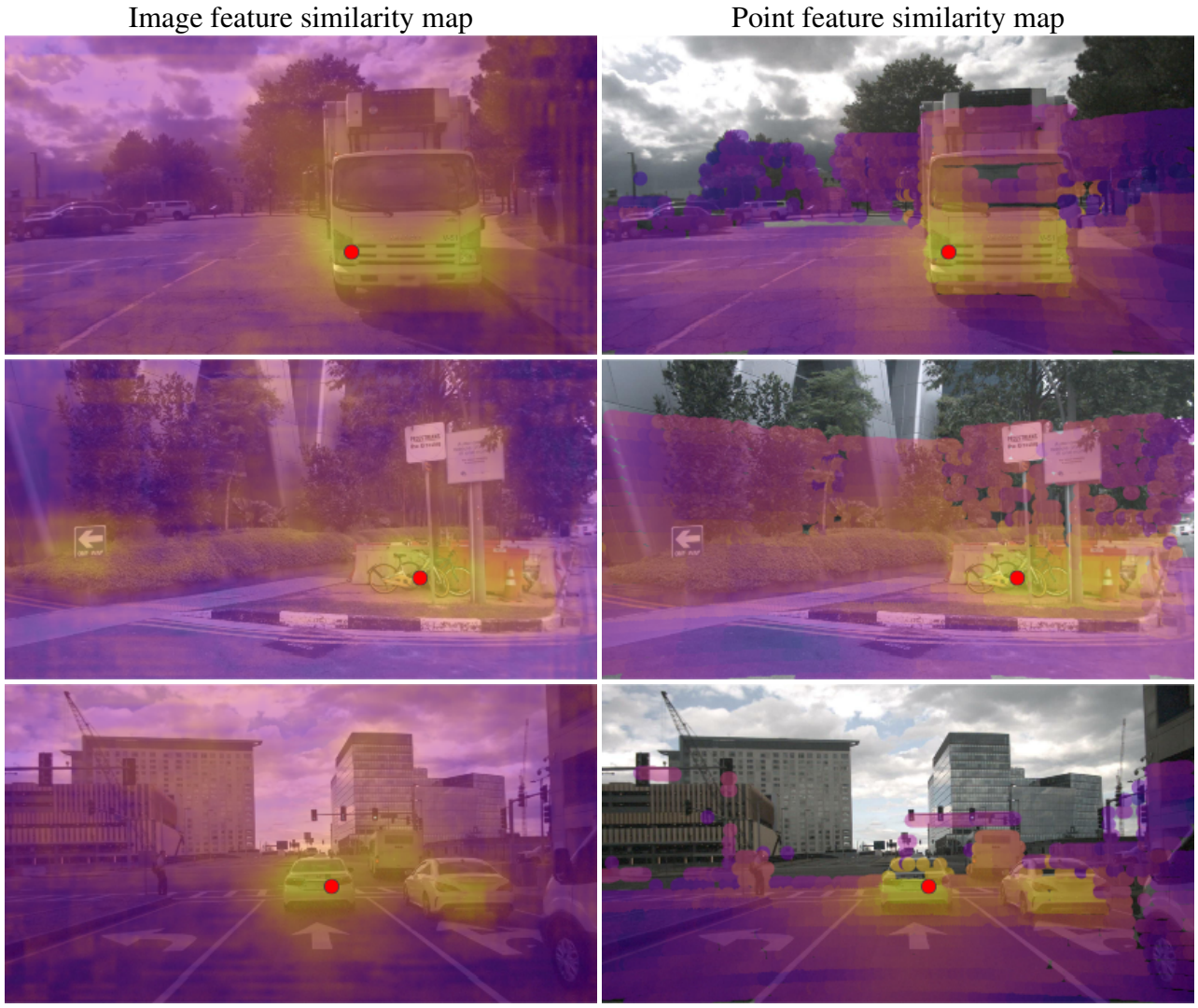

Figure 5. Cosine similarity between the SLidR's feature of a query point (displayed as a red dot) and: (a) the pixel features of an image in the same scene (left column - image feature similarity map); (b) the features of the other points projected in the same image (right column - point feature similarity map). The colormap goes from violet to yellow for respectively low and high similarity scores. We show these maps for two scenes in the validation set of [2].

els per image. Finally, we notice that SLidR is less sensitive to the choice of superpixel algorithm (e.g., using FH instead of SLIC) once its parameters are set correctly.

A.4. Few-Shot Semantic Segmentation

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

B. Data Augmentations

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

Regarding point clouds, we apply a random rotation around the z-axis and flip the direction of the x and y-axis with 50% probability for each axis. As in [10], we also drop points that lie in an axis-aligned random cuboid. Concretely, the cuboid center is placed on a randomly-selected point in the point cloud P and the length of each side covers at most 10% of the range of point coordinates on the

corresponding axis. We make sure that this dropped cuboid preserves at least 1024 pairs of points and pixels, otherwise another new cuboid is selected.

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

C. Baselines' Implementations Details

C.1. PPKT for Autonomous Driving.

PPKT [6] was originally proposed for RGB-D data captured indoor. As, up to now, there is no publicly released code, we propose our best adaption of this method in an autonomous driving setup, referred as PPKT†. PPKT† uses the same data augmentations, voxel-based 3D network backbone, and cylindrical coordinate voxels as our method.

PPKT† is obtained by: using $Q = M$ in Sec. 3, i.e., each superpixel contains one pixel; using strided convolutions in the image backbone $g_{\bar{\omega}_{\text{bck}}}(\cdot)$; the same image head $h_{\omega_{\text{head}}}$ as ours but with a bilinear upsamling layer from a resolution $1/32$ in each spatial direction to the original size of the input image. Furthermore, as computing the loss (3) is intractable when considering all possible point-pixel pairs $\mathcal{P} \in \{(\boldsymbol{f}_m^c, \boldsymbol{g}_m^c) : c = 1, \dots, C,\ m = 1, \dots, M\}$, a random subset of $\mathcal{P}$ is selected to compute it. In practice, as multiple scenes are available in a batch at train time, the set $\mathcal{P}$ also contains the point-pixel pairs of *all* scenes in our implementation. This sampling is not required in our method thanks to the use of superpixels which reduces the number of matching pairs. We use exactly the same parameters for pre-training a network with PPKT† or SLidR.

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

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

C.2. PointContrast

We retrained PointContrast [9] on nuScenes after studying several setups to optimize its performance. PointContrast requires pairs of point clouds acquired from different viewpoints in the same scene with a list of matching points in these two views. To provide a fair baseline, we tested different strategies to create this training dataset. Among the tested strategies, the best one consists in creating all possible pairs of keyframes within a scene, then removing pairs of point clouds which are less than 10 m apart, and removing those which have less than 1024 pairs of matching points.

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

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

Algorithm	None	SLIC [1]			FH [5]
#Superpixels		100	150	200	$\leqslant$143
mIoU	36.6	37.7	39.2	36.3	39.2

Table 6. Sensitivity of SLidR to the superpixel algorithms and superpixel parameters. The semantic segmentation scores (mIoU) are obtained by linear probing and computed on our nuScenes mini-val split. We compare the performance of SLidR when using (a) no superpixels; (b) SLIC [1] with different number of superpixels per image; and (c) the Felzenszwalb's [5] algorithm (FH) with parameters that produce at most 143 superpixels per image.

C.3. DepthContrast

The last baseline is DepthContrast [10] which pre-trains simultaneously two 3D network backbones, a point-based network, e.g., [7] and a voxel-based network, e.g., [11], using a contrastive task between the global point-cloud representations of the two networks. Among the three baselines, it is the only one which has already been used on a autonomous driving dataset: the Waymo Open Dataset [8]. To make it comparable with the rest of the methods in our study, we re-used the point-based network used in [10] while changing the voxel-based network to the same sparse residual U-Net that processes cylindrical coordinate vox-

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

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

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

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

els as in SLidR. After DepthContrast pre-training, we only evaluate on the downstream tasks the voxel-based network.

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

D. Additional Training Details

D.1. Linear Probing

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

D.2. Few-Shot Semantic Segmentation

On SemanticKITTI, the networks are fine-tuned for 100 epochs and a batch size of 10. On nuScenes, we use a batch size of 16 and for 100 epochs. We use different learning rates on the classification head and the backbone f_θ, except when the backbone is initialized with random weights. We recall that these learning rates are optimized for each method and each dataset, using our mini-val split for nuScenes and the validation set for semanticKITTI.

D.3. Annotation Efficiency on nuScenes

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

E. nuScenes mini-val Split

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

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

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

- 20 raining scenes,
- 12 night-time scenes,
- 44 scenes in Singapore,

- 56 scenes in Boston.

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

F. Societal and Environmental Impact

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

G. Public Resources Used

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

- KITTI object detection . . . . . CC BY-NC-SA 3.0
- MinkowskiEngine . . . . . MIT License
- nuScenes . . . . . CC BY-NC-SA 4.0
- nuScenes-devkit . . . . . Apache License 2.0
- OpenPCDet . . . . . Apache License 2.0
- Pytorch Lightning . . . . . Apache License 2.0
- Semantic KITTI . . . . . CC BY-NC-SA 4.0

References

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

[2] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *CVPR*, pages 11618–11628, 2020. 1, 2, 3

[3] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, pages 9650–9660, 2021. 1, 2

[4] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv:2003.04297*, 2020. 1

[5] Pedro F. Felzenszwalb and D. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 59:167–181, 2004. 2, 4

[6] 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. 4

[7] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, pages 5099–5108, 2017. 4

[8] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *CVPR*, pages 2443–2451, 2020. 4

[9] Saining Xie, Jiatao Gu, Demi Guo, Charles R. Qi, Leonidas J. Guibas, and Or Litany. PointContrast: Unsupervised pre-training for 3D point cloud understanding. In *ECCV*, pages 574–591, 2020. 4

[10] Zaiwei Zhang, Rohit Girdhar, Armand Joulin, and Ishan Misra. Self-supervised pretraining of 3D features on any point-cloud. In *ICCV*, pages 10252–10263, 2021. 3, 4

[11] Yin Zhou and Oncel Tuzel. VoxelNet: End-to-end learning for point cloud based 3D object detection. In *CVPR*, pages 4490–4499, 2018. 4