CLAP: Unsupervised 3D Representation Learning for Fusion 3D Perception via Curvature Sampling and Prototype Learning

Runjian Chen[1] Hang Zhang[2] Avinash Ravichandran[2] Wenqi Shao[4] Alex Wong[3*] Ping Luo[1*]

[1]The University of Hong Kong [2]Cruise [3]Yale University [4]Shanghai AI Laboratory

{rjchen, pluo}@cs.hku.hk {hang.zhang, avinash.ravichandran}@getcruise.com

alex.wong@yale.edu shaowenqi@pjlab.org.cn

Abstract

*Unsupervised 3D representation learning via masked-and-reconstruction with differentiable rendering is promising to reduce the labeling burden for fusion 3D perception. However, previous literature conduct pre-training for different modalities separately because of the hight GPU memory consumption. Consequently, the interaction between the two modalities (images and point clouds) is neglected during pre-training. In this paper, we explore joint unsupervised pre-training for fusion 3D perception via differentiable rendering and propose CLAP, short for **C**urvature samp**L**ing and sw**A**pping **P**rototype assignment prediction. The contributions are three-fold. 1) To overcome the GPU memory consumption problem, we propose Curvature Sampling to sample the more informative points/pixels for pre-training. 2) We propose to use learnable prototypes to represent parts of the scenes in a common feature space and bring the idea of swapping prototype assignment prediction to learn the interaction between the two modalities. 3) To further optimize learnable prototypes, we propose an Expectation-Maximization training scheme to maximize the similarity between embeddings and prototypes, followed by a Gram Matrix Regularization Loss to avoid collapse. Experiment results on NuScenes show that CLAP achieves 300% more performance gain as compared to previous SOTA 3D pre-training method via differentiable rendering. Codes and models will be released.*

1. Introduction

3D perception provides understanding of the ego vehicle's surrounding 3D space and plays an important role in downstream prediction and control tasks in autonomous driving. Single-modality 3D perception has long been studied with camera [6, 11, 19, 40, 45, 55] or LiDAR (Light-Detection-And-Ranging) [1, 12, 36, 39, 50, 54] inputs, which is less robust to changing lighting and weather conditions [30]. To compensate shortages of different modalities, research prevails on fusion 3D perception with camera images and LiDAR point clouds [7, 20, 21, 24, 25, 27, 31].

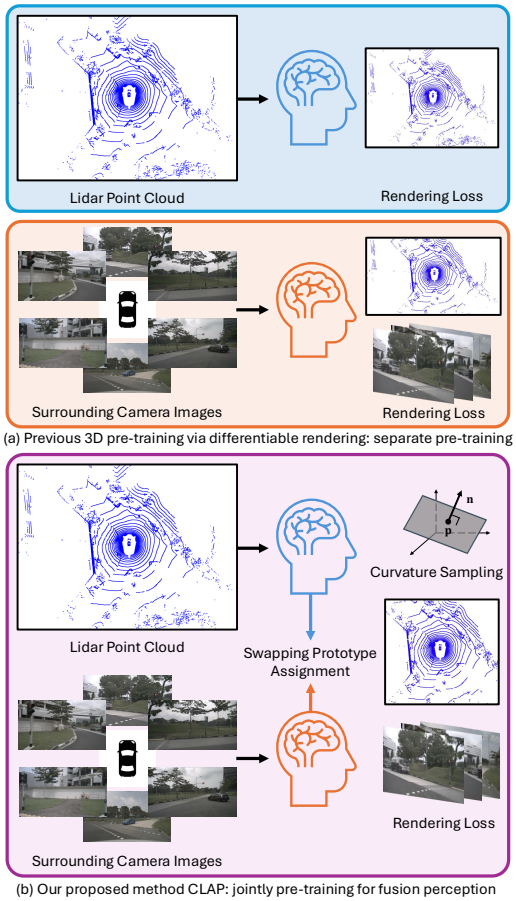

Figure 1. Unlike previous unsupervised 3D representation learning method UniPAD [53] that separately pre-train LiDAR and camera encoders (a), our proposed method CLAP conducts joint pre-training for fusion perception.

*Corresponding authors.

Meanwhile, labeling in 3D space is notoriously time-and-energy-consuming. Unsupervised 3D representation learning [5, 17, 18, 23, 48, 52, 57], which pre-trains backbones without any label and applies the pre-trained weight to initialize downstream models for performance improvement, shows potential to alleviate labeling burden in 3D perception. Hence, the community starts to explore pre-training for fusion 3D perception in an unsupervised manner [22, 41, 53, 56], among which UniPAD [53] shows superior performance. UniPAD [53] is a masked-reconstruction-based method and utilizes differentiable rendering as decoder during pre-training. However, during pre-training stage, the camera and LiDAR encoders are separately pre-trained without considering the interaction between the two modalities in [53]. The reason stems from the high GPU memory consumption. If all of the LiDAR points and camera pixels are considered for pre-training, the most advanced GPU up-to-date (H100) is only able to hold batchsize=1. But batchsize is an important hyperparameter for image encoder and a small one normally brings degradation to the performance, making it unfeasible for 3D pre-training.

In this paper, we explore joint unsupervised pre-training for fusion 3D perception via differentiable rendering and propose CLAP, which is shorthand for **C**urvature samp**L**ing and sw**A**pping **P**rototype assignment prediction. To begin with, we need to sample the LiDAR points and camera pixels for pre-training in order to make joint pre-training feasible. The most direct way is to use uniform sampling but as the sample number is small compared to the raw inputs ($\sim \frac{1}{100}$), uniform sampling does not brings improvement against separate pre-training. In order to sample more informative parts of the environment for pre-training, CLAP first estimates curvature of each LiDAR point in the 3D space by taking second order derivative of the SDF (signed distance field) function. Then we compute weights proportional to the estimated curvature and sample more informative points according to these weights. As for pixel sampling, we project the LiDAR points onto image planes and assign the weights to pixel. As LiDAR point clouds are sparse, we further apply gaussian blur to broadcast the weights to nearby pixels and sample pixels according to these weights. With Curvature Sampling, we further consider how to better utilize the interaction between camera images and LiDAR point clouds to gain some level of understanding about "objectness" or "parts of objects" in an unsupervised way. Inspired by SwAV [3], we propose to use learnable prototypes to learn to represent parts of the scenes. Similar to [3], CLAP applies a swapping prototype assignment prediction loss between camera and LiDAR embeddings to learn the common feature space. To further optimize the learnable prototypes, an Expectation-Maximization training scheme is proposed. In the Expectation step, we compute the probability that each prototype is assigned to the 3D embeddings in the 3D space. Then in the Maximization step, we maximize the similarity between embeddings and prototypes by minimizing the entropy of the assignment scores. Finally, to avoid collapse of the prototype [3], a gram matrix regularization Loss is proposed to minimize the correspondence among prototypes.

Through extensive experiments on the popular autonomous driving dataset NuScenes [2], we demonstrate that CLAP is able to achieve up to 300% more improvement than previous SOTA method [53].

2. Related Work

Fusion 3D Object Detection. Light-Detection-And-Ranging (LiDAR) and camera are important sensors for autonomous driving perception. Previous works mainly focus on single-modality 3D perception. For LiDAR-based 3D object detection, there are three main streams with different embedding schemes for point clouds inputs. 1) Embraced by [36, 38], point-based methods utilize point-level embeddings for 3D object detection. 2) Voxel-based methods [1, 10, 12, 50, 51, 54] voxelize the 3D scene and use sparse convolution or transformer for embedding. 3) Point-voxel-combined methods [37, 39] utilize both embeddings from 1) and 2). For camera-based 3D perception, [6, 11, 19, 40, 45, 55] embed 2D features on image plane and project these 2D features into 3D space with estimated depth. As single-modality perception can be degraded by various weather and lighting conditions [30], fusion 3D perception [7, 20, 21, 24, 25, 27, 31] starts to flourish. These methods mainly focus on the way to integrate embeddings of different modalities and use supervised pre-trained 2D and 3D backbones to achieve the SOTA performance. As labeling in 3D space is costly, we explore unsupervised 3D representation learning for fusion 3D perception.

3D Pre-training. Annotating for 3D data is notoriously time-and-energy-consuming and the emergence of unsupervised representation learning for 2D image [3, 13–15, 43, 46] provides a promising way to alleviate the annotation burden. Researcher starts to introduce unsupervised 3D representation learning into scene-level 3D point clouds [5, 16–18, 23, 26, 34, 47, 48, 52, 57], which can be divided into two contrastive-based and masked-and-reconstruction-based branches. Embraced by [5, 16, 18, 23, 26, 34, 47], contrastive-based works propose various ways to build suitable views and conduct contrastive learning to improve the performance in downstream perception task. Inspired by [15] in image domain, [17, 48, 52, 57] propose to first mask the input point clouds and pre-train the 3D encoders with a shallow decoder for reconstructing the unmasked inputs. [22, 41, 56] are pioneering works to introduce contrastive learning into fusion perception. They consider camera and

LiDAR embeddings as different views of the scene and apply the contrastive loss between the two modalities. Inspired by the success of neural field in representing 3D scenes [32, 44] and previous attempts to introduce neural rendering to 3D pre-training for point clouds [17, 49, 57], UniPAD [53] proposes to use a differentiable-rendering decoder for masked-and-reconstruction pre-training and achieves SOTA performance for unsupervised 3D representation learning on fusion 3D perception. However, due to the high GPU memory consumption, UniPAD [53] is only able to separately pre-train the image and point cloud encoders and fails to utilize the interaction between modalities during pre-training. In this paper, we explore unsupervised joint pre-training for 2D and 3D backbones via differentiable rendering.

3. Method

In this section, we introduce CLAP for unsupervised 3D pre-training via differentiable rendering on fusion 3D perception. As described in Fig. 2, CLAP pre-trains the image, LiDAR and fusion encoders jointly with Neural Field Rendering. In order to make joint pre-training feasible, Curvature Sampling is proposed as shown in Fig. 2 (a). To further make use of both modalities, we propose a swapping prototype assignment prediction scheme, which is visualized in Fig. 2 (b). We first discuss the formulation and overall pipeline in Section 3.1. Then we introduce the details about neural field and differentiable rendering in Section 3.2. Finally, we describe the curvature sampling and swapping prototype assignment prediction separately in Section 3.3 and 3.4.

3.1. Formulation and Pipeline

Notations. To begin with, we denote the input image set from N_{cam} cameras as $\mathcal{I} = \{\mathbf{I}_n \in \mathbb{R}^{H\times W\times 3}\}_{n=1}^{N_{\text{cam}}}$ and LiDAR point cloud as $\mathbf{P} \in \mathbb{R}^{N_p\times(3+d)}$. H and W are the height and width of the images and each pixel on the images has 3 values for RGB. N_p is the number of points in the LiDAR point cloud and each of them contains xyz-location and d feature channels. For example, in NuScenes [2] dataset, $d = 2$ represents the intensity and timestamp of each point and there are $N_{\text{cam}} = 6$ surrounding cameras on the autonomous vehicle. For each pair of camera image and LiDAR point cloud, we have the transformation matrix $\mathbf{T}_n \in \mathbb{R}^{3\times 4}$ indicating the projection between each camera plane and LiDAR coordinate, where $n = 1, 2, ..., N_{\text{cam}}$.

Encoding. The goal of unsupervised 3D representation learning for fusion perception is to pre-train the LiDAR, camera and fusion encoder in an unsupervised manner. Hence, we first voxelize and embed the raw LiDAR point cloud $\mathbf{P}$ with LiDAR encoder $f_{\text{P}}^{\text{enc}}$

$$\hat{\mathbf{P}} = f_{\text{P}}^{\text{enc}}(\mathbf{P}), \tag{1}$$

where $\hat{\mathbf{P}} \in \mathbb{R}^{\hat{D}\times\hat{H}\times\hat{W}\times\hat{d}_{\text{P}}}$ is the embedded 3D features for LiDAR point cloud. $\hat{D}$, $\hat{H}$ and $\hat{W}$ are spatial resolutions of the embedded features and $\hat{d}_{\text{P}}$ is number of feature channels after encoding. Then for camera images $\mathcal{I}$, we encode them and project the 2D features to 3D space with $\mathcal{T}$ with image encoder $f_{\text{I}}^{\text{enc}}$

$$\hat{\mathbf{I}} = f_{\text{I}}^{\text{enc}}(\mathcal{I}, \{\mathbf{T}_n\}_{n=1}^{N_{\text{cam}}}), \tag{2}$$

where $\hat{\mathbf{I}} \in \mathbb{R}^{\hat{D}\times\hat{H}\times\hat{W}\times\hat{d}_{\text{I}}}$ is the embedded 3D features for surrounding camera images with the similar dimensions as $\hat{\mathbf{P}}$ except for $\hat{d}_{\text{I}}$ feature channels. With $\hat{\mathbf{P}}$ and $\hat{\mathbf{I}}$, we further concatenate them along feature dimension and apply the fusion encoder $f_{\text{fusion}}^{\text{enc}}$ to get the fusion feature $\hat{\mathbf{F}} \in \mathbb{R}^{\hat{D}\times\hat{H}\times\hat{W}\times\hat{d}_{\text{F}}}$,

$$\hat{\mathbf{F}} = f_{\text{fusion}}^{\text{enc}}([\mathbf{P}, \mathbf{I}]). \tag{3}$$

Loss Function. To guide $f_{\text{P}}^{\text{enc}}$, $f_{\text{I}}^{\text{enc}}$ and $f_{\text{fusion}}^{\text{enc}}$ to learn good representations in an unsupervised manner, CLAP first embed the fusion features $\hat{\mathbf{F}}$ with a shallow 3D convolution network f^{3D} to get $\tilde{\mathbf{F}} = f^{\text{3D}}(\hat{\mathbf{F}})$ and we have $\tilde{\mathbf{F}} \in \mathbb{R}^{\hat{D}\times\hat{H}\times\hat{W}\times\hat{d}_{\text{F}}}$. Then a rendering loss $\mathcal{L}_{\text{rend}}$ for masked-reconstruction is applied on $\tilde{\mathbf{F}}$. Furthermore, a swapping prototype assignment prediction scheme $\mathcal{L}_{\text{proto}}$ is utilized in order to incorporate interaction of the two modalities into pre-training. The overall loss function is as below:

$$\mathcal{L} = \mathcal{L}_{\text{rend}}(\mathbf{P}, \tilde{\mathbf{F}}, \mathcal{T}) + \omega_{\text{proto}} \times \mathcal{L}_{\text{proto}}(\hat{\mathbf{P}}, \hat{\mathbf{I}}), \tag{4}$$

with ω_{proto} as a weighting parameter to balance the losses.

3.2. Neural Field and Differentiable Rendering

Inspired by the success of UniPAD [53], CLAP applies a differentiable rendering decoder with neural field to conduct the masked-and-reconstruction pre-training scheme. Here we first introduce Neural Field, which is the basis for camera images and point clouds rendering, and then discuss the differentiable rendering process on depth and RGB values.

Neural Field. Given a specific point $\mathbf{p} = [x, y, z] \in \mathbb{R}^3$ in the 3D space, the feature $\mathbf{f}_{\text{p}} \in \mathbb{R}^{\hat{d}_{\text{F}}}$ at $\mathbf{p}$ is queried from the fusion 3D embedding $\tilde{\mathbf{F}}$ by trilinear interpolation

$$\mathbf{f}_{\text{p}} = f^{\text{tri}}(\mathbf{p}, \hat{\mathbf{P}}_t), \tag{5}$$

where f^{tri} is an built-in module implemented in Pytorch [35]. Then we extract geometry features $\mathbf{f}_{\text{geo}} \in \mathbb{R}^{\hat{d}_{\text{geo}}}$ at $\mathbf{p}$ with f^{geo}, taking the concatenation of location $\mathbf{p}$ and queried feature $\mathbf{f}_{\text{p}}$ as inputs,

$$\mathbf{f}_{\text{geo}} = f^{\text{geo}}([\mathbf{p}, \mathbf{f}]). \tag{6}$$

With geometry features $\mathbf{f}_{\text{geo}}$, we predict the signed distance value $s \in \mathbb{R}$ [4, 28] and color value $s \in \mathbb{R}$ [44] at $\mathbf{p}$ with f^{SDF} and f^{RGB}. f^{geo}, f^{RGB} and f^{SDF} are parameterized by Multi-layer Perceptron,

$$s = f^{\text{SDF}}(\mathbf{f}_{\text{geo}}) \ , \ c = f^{\text{RGB}}(\mathbf{f}_{\text{geo}}). \tag{7}$$

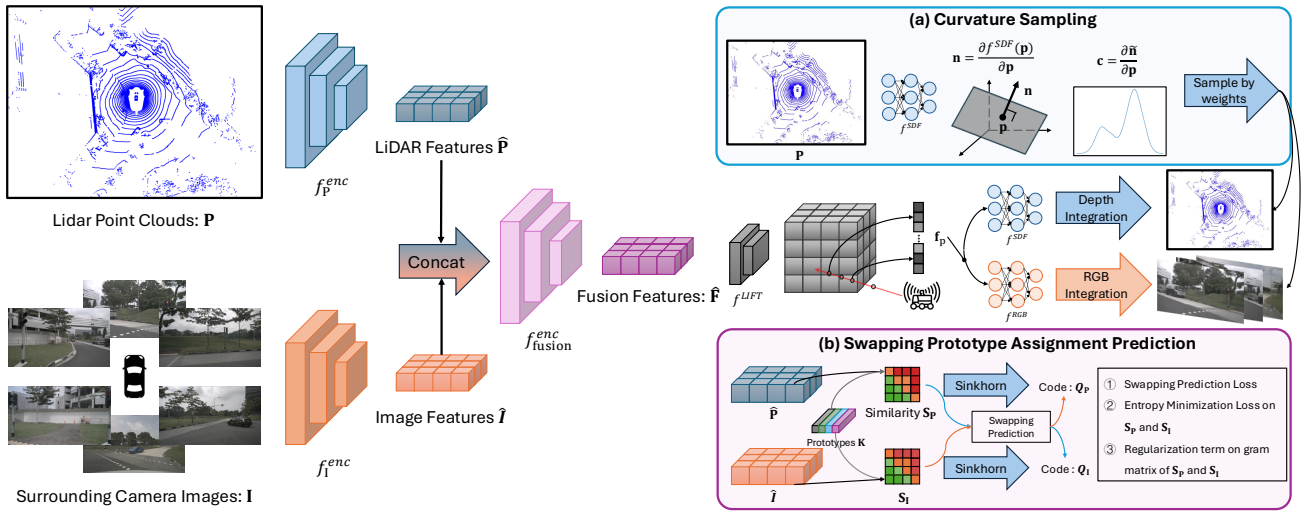

Figure 2. The pipeline of CLAP. In order to jointly pre-train the LiDAR, camera and fusion encoders, we first embed the paired LiDAR point clouds and camera images with $f_{\mathrm{P}}^{\mathrm{enc}}$, $f_{\mathrm{I}}^{\mathrm{enc}}$ and $f_{\mathrm{fusion}}^{\mathrm{enc}}$. Then based on the fusion features, CLAP applies differentiable rendering to predict both depth and rgb with the SDF and RGB values of the sampled points along LiDAR/camera rays from f^{SDF} and f^{RGB}, with which we compute loss against the observed LiDAR point cloud and camera images. To make joint pre-training feasible, we propose Curvature Sampling to sample informative parts of the 3D scene, as descriped in (a). Furthermore, we propose to use learnable prototypes to represent parts of objects and use a swapping prototype assignment prediction loss to incorporate the interaction of different modalities into pre-training. Finally, we optimize the learnable prototypes with an Expectation-Maximization scheme together with a gram matrix regularization loss.

Differentiable Rendering. Similar to [32, 44], we first sample N_{L} or N_{C} rays at the LiDAR or camera sensor origin $\mathbf{o}$, each of which is described by its normalized direction $\mathbf{d}$ and $\mathbf{o}$. Next, we sample N_{ray} points following [44] along each ray. Here each point along the ray can be interpreted by $\mathbf{p} = o + r\mathbf{d}$, where r is the range from the sensor origin to the point $\mathbf{p}$. Thus the sampled point set can be annotated by $\{\mathbf{p}_n = \mathbf{o} + r_n\mathbf{d}\}_{n=1}^{N_{\mathrm{ray}}}$ and we predict the estimated signed distance value s_n and color value c_n for them with f^{geo}, f^{RGB} and f^{SDF}. Following [44], we estimate the occupancy value α_n for each sampled point,

$$\alpha_n = \max\left(\frac{\Phi_h(s_n) - \Phi_h(s_{n+1})}{\Phi_h(s_n)}, 0\right). \tag{8}$$

Here $\Phi_h(x) = (1+e^{-hx})^{-1}$ stands for the sigmoid function paired with a learnable scalar h. After that, we predict the accumulated transmittance $\mathcal{T}_n$ similar to [44]

$$\mathcal{T}_n = \prod_{i=1}^{n-1}(1 - \alpha_i). \tag{9}$$

Based on $\mathcal{T}_n$, we compute an unbiased and occlusion-aware weight $w_n = \mathcal{T}_n\alpha_n$ [44] and integrate all samples along the ray to predict the range $\tilde{r}$ or color $\tilde{c}$ along this ray,

$$\tilde{r} = \sum_{n=1}^{N_{\mathrm{ray}}} w_n * r_n \ , \ \tilde{c} = \sum_{n=1}^{N_{\mathrm{ray}}} w_n * c_n \tag{10}$$

For the observed LiDAR points, it is evident that signed distance value at those points are 0. The final loss function for differentiable rendering is a combination of L-1 loss on range and color predictions and the surface SDF loss.

$$\mathcal{L}_{\mathrm{rend}} = \frac{1}{N_{\mathrm{L}}}\sum_{i=1}^{N_{\mathrm{L}}}(\omega_{\mathrm{L}}|r_i - \tilde{r}_i| + \omega_{\mathrm{sur}}|s_i|) + \omega_{\mathrm{C}}\frac{1}{N_{\mathrm{C}}}\sum_{i=1}^{N_{\mathrm{C}}}|c_i - \tilde{c}_i|, \tag{11}$$

where ω_{L}, ω_{sur} and ω_{C} are weighting parameters to balance the contributions from the three losses. s_i is the predicted signed distance value at the observed point along the sampled ray.

3.3. Curvature Sampling

In order to make joint unsupervised representation learning feasible, we have to make $N_{\mathrm{L}} \ll N_{\mathrm{P}}$ and $N_{\mathrm{C}} \ll H \cdot W \cdot N_{\mathrm{cam}}$. Intuitively, uniform sampling can be used. But due to the relatively small sample number compared to the raw inputs ($\sim \frac{1}{100}$), uniform sampling brings little improvement against separate pre-training, which is contradictory to our motivation. Hence, we need to sample more informative part of the scene for $\mathcal{L}_{\mathrm{rend}}$. As shown in Figure 3, we are inspired by the observation that surface with higher curvature (surface of a vehicle) generally contains more information as compared to that with lower curvature (road plane). Therefore, we propose the Curvature Sampling for

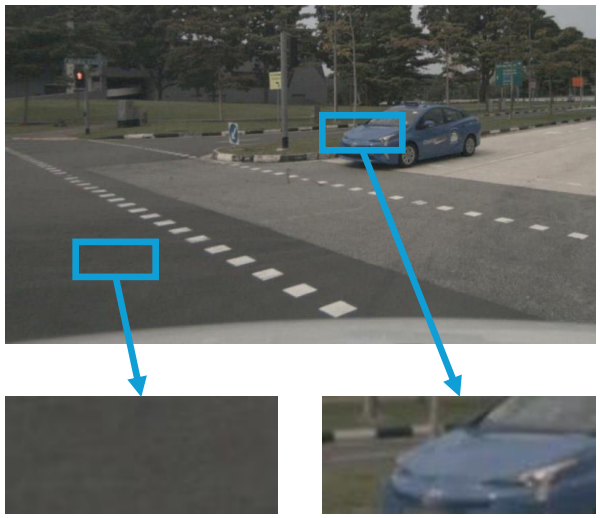

Figure 3. Inspiration of Curvature Sampling. From the two zoom-in areas, it can be found that areas with low curvature tend to be less informative, like road plane, while those with high curvature might provide more information for pre-training (objects like cars).

effective sampling. For each point $\mathbf{p}$ in the LiDAR point cloud $\mathbf{P}$, we first estimate the surface normal by deriving the signed distance function with respect to $\mathbf{p}$

$$\mathbf{n} = \frac{\delta f^{\text{SDF}}([\mathbf{p}, \mathbf{f}])}{\delta \mathbf{p}}, \tag{12}$$

where $\mathbf{n} \in \mathbb{R}^3$ is the predicted normal. Then we normalized $\mathbf{n}$ to get the direction of the normal

$$\tilde{\mathbf{n}} = \frac{\mathbf{n}}{||\mathbf{n}||_2}. \tag{13}$$

Here $||\cdot||_2$ is the L-2 norm. Next we estimate the mean curvature $\mathbf{c} \in \mathbb{R}^3$ at $\mathbf{p}$ by applying the vector differential operator on $\tilde{\mathbf{n}}$ with respect to $\mathbf{p}$

$$\mathbf{c} = \nabla_{\mathbf{p}} \tilde{\mathbf{n}}. \tag{14}$$

For each point $\mathbf{p}_n$ in LiDAR point cloud, we compute the average of $\mathbf{c}_n$ as the sampling weights ω_n and sample N_{L} LiDAR point cloud with a Multinomial Sampler implemented in PyTorch [35] for differentiable rendering. For pixels on image plane, we project the LiDAR point cloud back to image planes with $\mathcal{T}$, assign ω_n of each point to the projected pixel and apply a gaussian blur kernel of size K_{gaus} to densify the weights, with which we sample N_{C} pixel for $\mathcal{L}_{\text{rend}}$. As the estimation of curvature here is noisy especially during the first few epochs, we set up a warm-up stage with N_{warmup} epochs with uniform sampling and after N_{warmup} epochs, Curvature Sampling is utilized.

3.4. Swapping Prototype Assignment Prediction

CLAP aims to utilize the interaction between camera and LiDAR encoders for joint unsupervised 3D representation learning. Meanwhile, we hope to learn a common feature space standing for segments of different objects in an unsupervised manner. Inspired by SwAV [3], we propose to use learnable prototypes to represent parts of the scenes. We first randomly initialize N_{K} learnable prototypes $\mathbf{K} \in \mathbb{R}^{N_{\text{K}} \times d_{\text{K}}}$, each of which is a d_{K} vectors. Then we pre-train the backbones to understand the interaction between different modalities with a swapping prototype assignment prediction loss [3]. Furthermore, we introduce an Expectation-Maximization scheme to optimize the learnable prototypes and a gram matrix minimization loss to avoid collapse [3] of the prototype learning.

Swapping Prototype Assignment Prediction. We project the LiDAR embeddings $\hat{\mathbf{P}}$ and camera embeddings $\hat{\mathbf{I}}$ separately with two projection heads $f_{\text{P}}^{\text{proj}}$ and $f_{\text{I}}^{\text{proj}}$ to the same dimension as $\mathbf{K}$

$$\dot{\mathbf{P}} = f_{\text{P}}^{\text{proj}}(\hat{\mathbf{P}}) \ , \ \dot{\mathbf{I}} = f_{\text{I}}^{\text{proj}}(\hat{\mathbf{I}}), \tag{15}$$

where $f_{\text{P}}^{\text{proj}}$ and $f_{\text{I}}^{\text{proj}}$ are parameterized by Multi-Layer Perceptron and $\dot{\mathbf{P}} \in \mathbb{R}^{\hat{D} \times \hat{H} \times \hat{W} \times \hat{d}_{\text{K}}}$, $\dot{\mathbf{I}} \in \mathbb{R}^{\hat{D} \times \hat{H} \times \hat{W} \times \hat{d}_{\text{K}}}$. We denote $N_{\text{3D}} = \hat{D} \times \hat{H} \times \hat{W}$ and then normalize and reshape the projected embeddings into $\dot{\mathbf{P}} \in \mathbb{R}^{N_{\text{3D}} \times \hat{d}_{\text{K}}}$ and $\dot{\mathbf{I}} \in \mathbb{R}^{N_{\text{3D}} \times \hat{d}_{\text{K}}}$. After that, similarity scores $\mathbf{S}_{\text{P/I}} \in \mathbb{R}^{N_{\text{3D}} \times N_{\text{K}}}$ between 3D embeddings and prototypes are computed separately for LiDAR and camera branches

$$\mathbf{S}_{\text{P}} = \dot{\mathbf{P}} \cdot \mathbf{K}^{\intercal} \ , \ \mathbf{S}_{\text{I}} = \dot{\mathbf{I}} \cdot \mathbf{K}^{\intercal}. \tag{16}$$

Similar to SwAV [3], we detach $\mathbf{S}_{\text{P/I}}$ and apply sinkhorn algorithm [9] to make them approach the double stochastic matrix in N_{sink} iterations. We denote the updated matrix as codes $\mathbf{Q}_{\text{P/I}} \in \mathbb{R}^{N_{\text{3D}} \times N_{\text{K}}}$. The swapping prototype assignment loss is formulated with a temperature parameter τ [3],

$$\begin{aligned}\mathcal{L}_{\text{SwAV}} = -\frac{1}{N_{\text{3D}} N_{\text{K}}} \sum_{n=1}^{N_{\text{3D}}} \sum_{m=1}^{N_{\text{K}}} \{ \mathbf{Q}_{\text{I}}^{n,m} \log \frac{exp(\mathbf{S}_{\text{P}}^{n,m})/\tau}{\sum_{k=1}^{N_{\text{K}}} exp(\mathbf{S}_{\text{P}}^{n,k})/\tau} \\ + \mathbf{Q}_{\text{P}}^{n,m} \log \frac{exp(\mathbf{S}_{\text{I}}^{n,m})/\tau}{\sum_{k=1}^{N_{\text{K}}} exp(\mathbf{S}_{\text{I}}^{n,k})/\tau} \}.\end{aligned} \tag{17}$$

Expectation-Maximization. In order to guide the learnable prototypes to represent parts of the environment, we introduce the Expectation-Maximization algorithm [33] to further optimize the prototypes. In the Expectation step, we compute the probability $\hat{\mathbf{S}}_{\text{P/I}}$ that each prototype is assigned to each embeddings by applying a softmax operation on $\mathbf{S}_{\text{P/I}}$. Then for Maximization step, we expect to maximize the probability of the assignment between one prototype to

one specific part of the scene and this is the same as minimizing the entropy of the similarity matrix. Hence, the EM loss is computed as,

$$\mathcal{L}_{\text{EM}} = -\frac{1}{N_{\text{3D}} N_{\text{K}}} \sum_{n=1}^{N_{\text{3D}}} \sum_{m=1}^{N_{\text{K}}} \{\hat{\mathbf{S}}_{\text{P}}^{n,m} log \hat{\mathbf{S}}_{\text{P}}^{n,m} + \hat{\mathbf{S}}_{\text{I}}^{n,m} log \hat{\mathbf{S}}_{\text{I}}^{n,m}\} \tag{18}$$

Gram Matrix Minimization. When training the randomly initialized prototypes, the network might learn a short cut with all prototypes being the same [3], which is called collapse. To avoid this, we estimate similarity between prototypes by the gram matrix $\mathbf{G} = \mathbf{K}\mathbf{K}^{\intercal}$ of prototypes $\mathbf{K}$, the dimension of which is $\mathbf{G} \in \mathbb{R}^{N_{\text{K}} \times N_{\text{K}}}$. Finally we minimize the average of the non-diagonal elements of $\mathbf{G}$ in order to avoid collapse

$$\mathcal{L}_{\text{GMM}} = \frac{1}{N_{\text{K}}(N_{\text{K}} - 1)} \sum_{n}^{N_{\text{K}}} \sum_{m=1, m \neq n}^{N_{\text{K}}} \mathbf{G}^{n,m}. \tag{19}$$

Overall Prototype Learning Loss. We apply weighting parameters ω_{SwAV}, ω_{EM} and ω_{GMM} to balance the three losses proposed above, which leads to the overall loss function for prototype learning,

$$\mathcal{L}_{\text{proto}} = \omega_{\text{SwAV}} \mathcal{L}_{\text{SwAV}} + \omega_{\text{EM}} \mathcal{L}_{\text{EM}} + \omega_{\text{GMM}} \mathcal{L}_{\text{GMM}}. \tag{20}$$

4. Experiments

Unsupervised 3D representation learning for fusion perception aims to pre-train both LiDAR and camera encoders and initialize downstream models with the pre-trained weights to gain performance improvement in downstream tasks. In this section, we design extensive experiments on the popular autonomous driving dataset NuScenes [2] to demonstrate the effectiveness of CLAP. To begin with, we describe experiment setups in Section 4.1. Next, we show and analyze main results in Section 4.2. Finally, we provide ablation study and visualizations separately in Section 4.3 and 4.4.

4.1. Settings

Datasets. We use the popular autonomous driving dataset NuScenes [2] to evaluate the performance of CLAP. NuScenes [2] uses one roof LiDAR and six surrounding cameras to collect data. The LiDAR is a 32-beam Velodyne and collecting frequency is 20Hz. The frequency of camera capturing is 12Hz. [2] conducts the synchronization and provides paired data of LiDAR point cloud and camera images. The whole NuScenes dataset contains 1000 scenes collected in Boston and Singapore. Each scene lasts for around 20 seconds and there are a total of 5.5 hours data. Following convention practice from [2, 42], we divide the whole dataset into training set with 850 scenes and validation set with 150 scenes. Pre-trainings are conducted on the whole training set and downstream task is 3D object detection under few-shot setting, where we randomly sample 5% labeled data in the training set to train the randomly initialized model and fine-tune the pre-trained ones. The overall mAP (mean accurate precisions) and NDS (NuScenes Detection Score) are reported, along with AP (accurate precisions) for different categories.

Downstream 3D Object Detectors. We conduct experiments on one of the SOTA fusion 3D object detector called BEVFusion [27]. BEVFusion is implemented in the popular code repository for autonomous driving perception called OpenPCDet [42]. We use average precisions of various categories (APs), mean average precision (mAP) and NuScenes Detection Score (NDS) [2] as the evaluation metrics. We follow a similar setting in [48] to gradually increase training iterations of the model without pre-trained weights until convergence is observed. Here convergence means further increasing training iterations will not improve the performance. Then the number of training iterations is fixed for fine-tuning pre-trained models. This setting ***avoid the case that pre-training only accelerate convergence and make sure that pre-training indeed improve the performance of downstream models***, that is improving the sample efficiency of the downstream task.

Baseline Pre-training Method for Fusion Perception. We incorporate UniPAD [53] as the baseline 3D unsupervised pre-training method. We utilize the official implementation from [53] to pre-train the backbones of BEVFusion [27]. As pre-trained image backbones are broadly used for image feature extraction, the implementation of [53] also use a pre-trained image backbone from [8] to initiate their pre-training and the training set for this backbone does not involve any data from NuScenes [2]. We adopt this practice and also for training-from-scratch model, we initialize the camera encoder with the same pre-trained weights from [8].

Implementation Details of CLAP. The feature channels for LiDAR embeddings $\hat{\mathbf{P}}$, image embeddings $\hat{\mathbf{I}}$, fusion embeddings $\hat{\mathbf{F}}$ and swapping prototype assignment prediction are respectively set to $\hat{d}_{\text{P}} = 256$, $\hat{d}_{\text{I}} = 80$, $\hat{d}_{\text{F}} = 512$ and $\hat{d}_{\text{K}} = 128$. Sampling number for LiDAR point cloud and camera pixel are $N_{\text{L}} = 8192$ and $N_{\text{C}} = 1024 \times N_{\text{cam}}$, where $N_{\text{cam}} = 6$ for NuScenes dataset [2]. The number of sample points along each ray is $N_{\text{ray}} = 96$. Warm-up epochs for Curvature Sampling is $N_{\text{warmup}} = 4$. For swapping prototype assignment prediction, we set the number of learnable prototypes and sinkhorn update iterations to $N_{\text{K}} = 512$ and $N_{\text{sink}} = 3$. The temperature for swapping prediction loss is $\tau = 1.0$. The loss weighting parameters are implemented as $\omega_{\text{proto}} = 1.0$, $\omega_{\text{L}} = 2.0$, $\omega_{\text{sur}} = 0.1$, $\omega_{\text{C}} = 0.1$, $\omega_{\text{SwAV}} = 1.0$, $\omega_{\text{EM}} = 0.1$ and $\omega_{\text{GMM}} = 0.1$. We use a learning rate of

Init.	mAP	NDS	Car	Truck	C.V.	Bus	Trailer	Barrier	Mot.	Bic.	Ped.	T.C.
Rand.	48.69	55.28	78.52	46.64	16.18	50.44	22.11	57.00	46.87	30.56	76.16	62.40
UniPAD	49.81 $^{+0.72}$	55.29 $^{+0.01}$	80.81	42.81	17.08	48.98	25.85	61.72	50.19	27.53	78.53	64.57
CLAP	**51.17** $^{+2.48}$	**57.04** $^{+1.76}$	79.56	48.43	18.84	56.34	23.98	60.60	48.87	34.11	78.08	62.87

Table 1. Results for fine-tuning on 5% of training set in NuScenes [2]. "Init." means the way to initialize models, including the baseline UniPAD [53]. To avoid the case the performance difference is brought by accelerating convergence, we first increase the training iteration of training-from-scratch model (BEVFusion [27]) until we observe convergence, which leads to the "Rand." row. Then we pre-train the backbones with UniPAD [53] and the proposed CLAP, initialize BEVFusion with the pre-trained weights and then fine-tune the model with the same number iterations as "Rand.". We provide mAP and NDS as an evaluation of the overall performance of different models and highlight the best mAP and NDS with bold font. We also indicate the performance improvement by green color. For detailed categories, we provide AP for them and "C.V.", "Mot.", "Bic.", "Ped." and "T.C." are abbreviations for Construction Vehicle, Motorcycle, Bicycle, Pedestrian and Traffic Cone. All the results here are in %.

0.00005 with a cosine learning schedule for pre-training and use mask augmentation for CLAP with a masking rate of 0.9.

4.2. Main Results

Both CLAP and baseline methods are pre-trained on the whole training set of NuScenes [2] without labels. To compare the effectiveness of CLAP and UniPAD [53], we randomly sample 5% of the training labels and fine-tune BEVFusion [27] initialized by the pre-trained weights. Furthermore, as the size of pre-training dataset is limited, we randomly sample 2.5%, 1% and 0.5% of the training labels and fine-tune BEVFusion with CLAP pre-trained weights to see whether potential scaling property exists in pre-training with CLAP.

Comparison to Previous SOTA Method. As shown in Table 1, CLAP achieves 2.48% mAP improvement over randomly initialization at convergence, which is 300% more improvement for mAP than SOTA unsupervised 3D representation method UniPAD [53]. For NDS metric, UniPAD [53] only achieves comparable performance while CLAP surpasses the train-from-scratch model by 1.76% . When it turns to different categories, CLAP generally benefit the performance of all the categories and for Construction Vehicle, Bus, Barrier, Motorcycle and Bicycle, the improvement over random initialization are more than 2% AP.

Potential Scaling Property. As we are not able to scale up the pre-training dataset at current stage, we explore potential scaling property by gradually decreasing the sample numbers (2.5%, 1% and 0.5%) for fine-tuning, which increases the ratio between pre-training data and fine-tuning data. The results are shown in Table 2 together with the results of fine-tuning on 5% of training set in NuScenes. It can be found that as the ratio between pre-training data and fine-tuning data gets larger, the performance improvement by CLAP increases and CLAP provides a gain up to 7.22% mAP and 4.71% NDS with 0.5% fine-tuning data. These results show that CLAP is promising in scaling property and in the future, if we can scale up the pre-training dataset, CLAP might further improve current SOTA performance.

Init.	Fine-tune Data	mAP	NDS
Random	5%	48.69	55.28
CLAP		51.17 $^{+2.48}$	57.04 $^{+1.76}$
Random	2.5%	39.12	40.01
CLAP		42.86 $^{+3.74}$	42.18 $^{+2.17}$
Random	1%	26.22	29.82
CLAP		30.53 $^{+4.31}$	31.87 $^{+2.05}$
Random	0.5%	16.49	22.61
CLAP		23.71 $^{+7.22}$	27.32 $^{+4.71}$

Table 2. Results on potential scaling property of CLAP . "Init." means the method for initialization. "Fine-tune Data" stands for the sample number of training set in NuScenes [2] used for fine-tuning/training-from-scratch. We gradually increase the number of training iterations for training-from-scratch model (BEVFusion [27]) with different numbers of training data until we observe convergence, leading to the results of "Rand.". Then we initialize BEVFusion with weights pre-trained by CLAP and then fine-tune the model with the same numbers of iterations as "Rand.". We provide mAP and NDS as an evaluation of the overall performance. All the results here are in %.

4.3. Visualizations

In this section, we provide visualization to show the effectiveness of the proposed Curvature Sampling. In order to evaluate whether the estimated curvature is able to help sampling more informative part of the environment, we use the pre-trained model to estimate the curvature of LiDAR point clouds and use heatmap color to indicate the weight computed in Section 3.3. The visualization results are shown in Figure 4. We use orange boxes to highlight those regions with relatively correct estimation and green ones for those with noisy estimation. Furthermore, for bet-

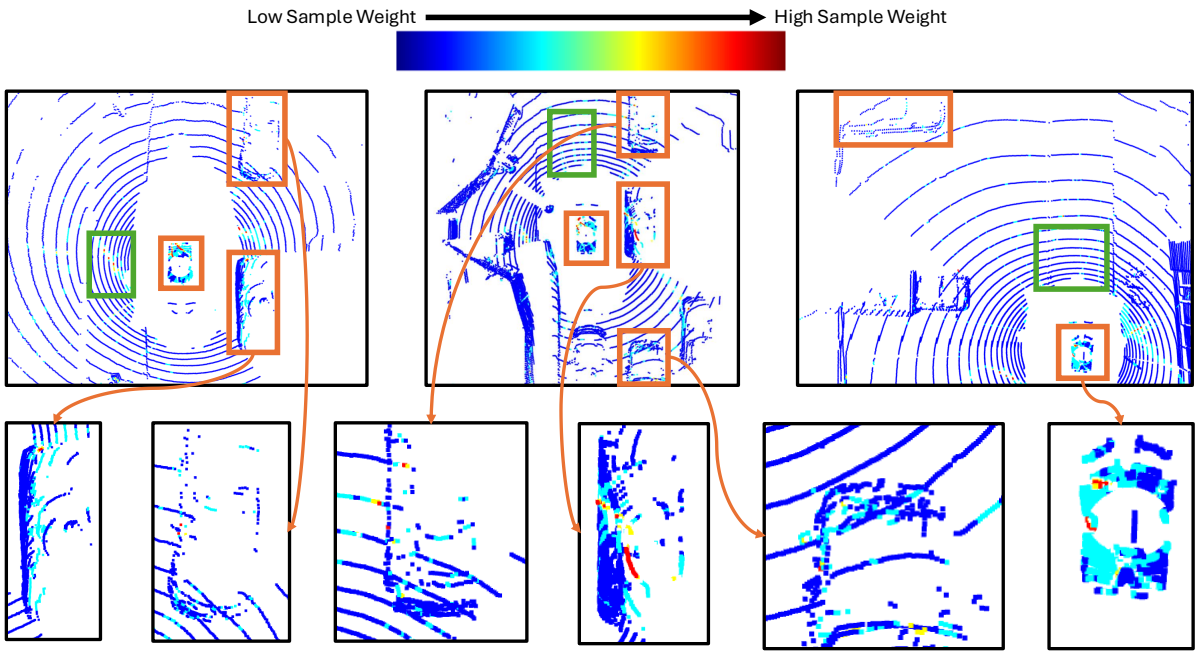

Figure 4. Visualization of curvature estimation. The color of the points change from blue to red, indicating lower sample weight / curvature to higher ones. We highlight those correct regions with orange boxes and those noisy estimation with green ones. Some of the correct estimated regions are further zoom-in for better understanding. Best view in color.

Joint Pre-train	Cur. Sam.	Proto. Learning	mAP
✗	✗	✗	49.81
✓	✗	✗	49.55
✓	✓	✗	50.81
✓	✓	✓	51.17

Table 3. Results for ablation study. We use BEVFusion [27] as the downstream models. Results are mAPs in %. The first line is seperate pre-training with UniPAD [53] The second one is jointly pre-training using UniPAD [53] and random sampling to address the GPU memory limitation. Then we subsequently add Curvature Sampling and Prototype Learning, which are results in third and fourth lines. Results are mAPs in %.

ter understanding about the curvature estimation, we zoom in for some correctly estimated regions. It can be found that though some noise exists, CLAP is able to predict high weights for those highly informative region for sampling and meanwhile assign lower weights to most of the background, which makes joint pre-training feasible.

4.4. Ablation Study

We conduct ablation study to evaluate the effectiveness of different components. As shown in Table 3, it can be found that using uniform sampling (second line) to make joint pre-training feasible does not bring improvement over separate pre-training (first line) [53]. Then, Curvature Sampling (third line) improves the performance over uniform sampling and separate pre-training by assigning larger weights to more informative parts of the 3D scene during sampling. Finally, the Prototype Learning scheme (fourth line) introduces interaction of LiDAR and camera encoders into pre-training and achieve the best performance.

5. Conclusion

In this paper, we propose CLAP to jointly pre-train for fusion perception in an unsupervised manner via differentiable rendering. To make joint pre-training feasible, CLAP utilizes the novel Curvature Sampling to sample more informative parts of the 3D scene for pre-training. Further, a swapping prototype prediction scheme is introduced in order to make full use of the interaction between LiDAR point clouds and camera images. To make the learnable prototypes better represent parts/semantics of objects, CLAP brings the idea of Expectation-Maximization for prototypes optimization, alongside with a gram matrix regularization term to avoid collapse. Experiment results demonstrate that CLAP is superior in unsupervised 3D representation learning and has the potential to scale up. In the future, when a larger pre-training dataset is available, we expect to scale up the pre-training with CLAP for more performance improvement.

References

[1] Xuyang Bai, Zeyu Hu, Xinge Zhu, Qingqiu Huang, Yilun Chen, Hongbo Fu, and Chiew-Lan Tai. Transfusion: Robust lidar-camera fusion for 3d object detection with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1090–1099, 2022. 1, 2

[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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11621–11631, 2020. 2, 3, 6, 7, 1

[3] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *Advances in Neural Information Processing Systems*, pages 9912–9924. Curran Associates, Inc., 2020. 2, 5, 6

[4] Tony Chan and Wei Zhu. Level set based shape prior segmentation. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, pages 1164–1170. IEEE, 2005. 3

[5] Runjian Chen, Yao Mu, Runsen Xu, Wenqi Shao, Chenhan Jiang, Hang Xu, Zhenguo Li, and Ping Luo. Coˆ 3: Cooperative unsupervised 3d representation learning for autonomous driving. *arXiv preprint arXiv:2206.04028*, 2022. 2

[6] Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. Monocular 3d object detection for autonomous driving. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2147–2156, 2016. 1, 2

[7] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1907–1915, 2017. 1, 2

[8] MMDetection3D Contributors. MMDetection3D: OpenMMLab next-generation platform for general 3D object detection. https://github.com/open-mmlab/mmdetection3d, 2020. 6

[9] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013. 5

[10] Jiajun Deng, Shaoshuai Shi, Peiwei Li, Wengang Zhou, Yanyong Zhang, and Houqiang Li. Voxel r-cnn: Towards high performance voxel-based 3d object detection. In *Proceedings of the AAAI conference on artificial intelligence*, pages 1201–1209, 2021. 2

[11] Mingyu Ding, Yuqi Huo, Hongwei Yi, Zhe Wang, Jianping Shi, Zhiwu Lu, and Ping Luo. Learning depth-guided convolutions for monocular 3d object detection. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition workshops*, pages 1000–1001, 2020. 1, 2

[12] Lue Fan, Ziqi Pang, Tianyuan Zhang, Yu-Xiong Wang, Hang Zhao, Feng Wang, Naiyan Wang, and Zhaoxiang Zhang. Embracing single stride 3d object detector with sparse transformer. *arXiv preprint arXiv:2112.06375*, 2021. 1, 2

[13] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, koray kavukcuoglu, Remi Munos, and Michal Valko. Bootstrap your own latent - a new approach to self-supervised learning. In *Advances in Neural Information Processing Systems*, pages 21271–21284. Curran Associates, Inc., 2020. 2

[14] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

[15] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022. 2

[16] Ji Hou, Benjamin Graham, Matthias Nießner, and Saining Xie. Exploring data-efficient 3d scene understanding with contrastive scene contexts. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15587–15597, 2021. 2

[17] Di Huang, Sida Peng, Tong He, Honghui Yang, Xiaowei Zhou, and Wanli Ouyang. Ponder: Point cloud pre-training via neural rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16089–16098, 2023. 2, 3

[18] Siyuan Huang, Yichen Xie, Song-Chun Zhu, and Yixin Zhu. Spatio-temporal self-supervised representation learning for 3d point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6535–6545, 2021. 2

[19] Jason Ku, Alex D Pon, and Steven L Waslander. Monocular 3d object detection leveraging accurate proposals and shape reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11867–11876, 2019. 1, 2

[20] Yanwei Li, Xiaojuan Qi, Yukang Chen, Liwei Wang, Zeming Li, Jian Sun, and Jiaya Jia. Voxel field fusion for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1120–1129, 2022. 1, 2

[21] Yingwei Li, Adams Wei Yu, Tianjian Meng, Ben Caine, Jiquan Ngiam, Daiyi Peng, Junyang Shen, Yifeng Lu, Denny Zhou, Quoc V Le, et al. Deepfusion: Lidar-camera deep fusion for multi-modal 3d object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 17182–17191, 2022. 1, 2

[22] 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 *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1500–1508, 2022. 2

[23] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3293–3302, 2021. 2

[24] Ming Liang, Bin Yang, Yun Chen, Rui Hu, and Raquel Urtasun. Multi-task multi-sensor fusion for 3d object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7345–7353, 2019. 1, 2

[25] Tingting Liang, Hongwei Xie, Kaicheng Yu, Zhongyu Xia, Zhiwei Lin, Yongtao Wang, Tao Tang, Bing Wang, and Zhi Tang. Bevfusion: A simple and robust lidar-camera fusion framework. *Advances in Neural Information Processing Systems*, 35:10421–10434, 2022. 1, 2

[26] Yunze Liu, Li Yi, Shanghang Zhang, Qingnan Fan, Thomas Funkhouser, and Hao Dong. P4contrast: Contrastive learning with pairs of point-pixel pairs for rgb-d scene understanding. *arXiv preprint arXiv:2012.13089*, 2020. 2

[27] Zhijian Liu, Haotian Tang, Alexander Amini, Xinyu Yang, Huizi Mao, Daniela L Rus, and Song Han. Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation. In *2023 IEEE international conference on robotics and automation (ICRA)*, pages 2774–2781. IEEE, 2023. 1, 2, 6, 7, 8

[28] Ravi Malladi, James A Sethian, and Baba C Vemuri. Shape modeling with front propagation: A level set approach. *IEEE transactions on pattern analysis and machine intelligence*, 17(2):158–175, 1995. 3

[29] Jiageng Mao, Minzhe Niu, Chenhan Jiang, Hanxue Liang, Jingheng Chen, Xiaodan Liang, Yamin Li, Chaoqiang Ye, Wei Zhang, Zhenguo Li, et al. One million scenes for autonomous driving: Once dataset. *arXiv preprint arXiv:2106.11037*, 2021. 1, 2

[30] Jiageng Mao, Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. 3d object detection for autonomous driving: A comprehensive survey. *International Journal of Computer Vision*, 131(8):1909–1963, 2023. 1, 2

[31] Gregory P Meyer, Jake Charland, Darshan Hegde, Ankit Laddha, and Carlos Vallespi-Gonzalez. Sensor fusion for joint 3d object detection and semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 0–0, 2019. 1, 2

[32] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 3, 4

[33] Todd K Moon. The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6):47–60, 1996. 5

[34] Bo Pang, Hongchi Xia, and Cewu Lu. Unsupervised 3d point cloud representation learning by triangle constrained contrast for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5229–5239, 2023. 2

[35] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 3, 5

[36] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2

[37] 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10529–10538, 2020. 2

[38] Shaoshuai Shi, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network. *IEEE transactions on pattern analysis and machine intelligence*, 43(8):2647–2664, 2020. 2

[39] Shaoshuai Shi, Li Jiang, Jiajun Deng, Zhe Wang, Chaoxu Guo, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn++: Point-voxel feature set abstraction with local vector representation for 3d object detection. *arXiv preprint arXiv:2102.00463*, 2021. 1, 2

[40] Andrea Simonelli, Samuel Rota Bulo, Lorenzo Porzi, Manuel López-Antequera, and Peter Kontschieder. Disentangling monocular 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1991–1999, 2019. 1, 2

[41] Jiachen Sun, Haizhong Zheng, Qingzhao Zhang, Atul Prakash, Z Morley Mao, and Chaowei Xiao. Calico: Self-supervised camera-lidar contrastive pre-training for bev perception. *arXiv preprint arXiv:2306.00349*, 2023. 2

[42] OpenPCDet Development Team. Openpcdet: An open-source toolbox for 3d object detection from point clouds. https://github.com/open-mmlab/OpenPCDet, 2020. 6, 2

[43] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019. 2

[44] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *arXiv preprint arXiv:2106.10689*, 2021. 3, 4

[45] Tai Wang, Xinge Zhu, Jiangmiao Pang, and Dahua Lin. Fcos3d: Fully convolutional one-stage monocular 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 913–922, 2021. 1, 2

[46] Xinlong Wang, Rufeng Zhang, Chunhua Shen, Tao Kong, and Lei Li. Dense contrastive learning for self-supervised visual pre-training. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[47] Saining Xie, Jiatao Gu, Demi Guo, Charles R Qi, Leonidas Guibas, and Or Litany. Pointcontrast: Unsupervised pre-training for 3d point cloud understanding. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 574–591. Springer, 2020. 2

[48] Runsen Xu, Tai Wang, Wenwei Zhang, Runjian Chen, Jinkun Cao, Jiangmiao Pang, and Dahua Lin. Mv-jar: Masked voxel jigsaw and reconstruction for lidar-based self-supervised pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13445–13454, 2023. 2, 6

[49] Siming Yan, Zhenpei Yang, Haoxiang Li, Chen Song, Li Guan, Hao Kang, Gang Hua, and Qixing Huang. Implicit autoencoder for point-cloud self-supervised representation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14530–14542, 2023. 3

[50] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. 1, 2

[51] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7652–7660, 2018. 2

[52] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9403–9414, 2023. 2

[53] Honghui Yang, Sha Zhang, Di Huang, Xiaoyang Wu, Haoyi Zhu, Tong He, Shixiang Tang, Hengshuang Zhao, Qibo Qiu, Binbin Lin, et al. Unipad: A universal pre-training paradigm for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15238–15250, 2024. 1, 2, 3, 6, 7, 8

[54] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11784–11793, 2021. 1, 2

[55] Yunpeng Zhang, Jiwen Lu, and Jie Zhou. Objects are different: Flexible monocular 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3289–3298, 2021. 1, 2

[56] Yifan Zhang, Siyu Ren, Junhui Hou, Jinjian Wu, Yixuan Yuan, and Guangming Shi. Self-supervised learning of lidar 3d point clouds via 2d-3d neural calibration. *arXiv preprint arXiv:2401.12452*, 2024. 2

[57] Haoyi Zhu, Honghui Yang, Xiaoyang Wu, Di Huang, Sha Zhang, Xianglong He, Tong He, Hengshuang Zhao, Chunhua Shen, Yu Qiao, et al. Ponderv2: Pave the way for 3d foundataion model with a universal pre-training paradigm. *arXiv preprint arXiv:2310.08586*, 2023. 2, 3

CLAP: Unsupervised 3D Representation Learning for Fusion 3D Perception via Curvature Sampling and Prototype Learning

Supplementary Material

A. Transferring to other datasets

We conduct further experiments to evaluate the transferring ability of CLAP. Specifically, we select LiDAR-based 3D object detector CenterPoint [54] on Once [29] for the downstream task. A 40-beam LiDAR is utilized in Once [29] to collect 15k labeled training data. We randomly sample 5% and also use all of the labeled training set to train the from-scratch model until convergence is observed. Then we use pre-trained weights by CLAP on NuScenes [2] to initialize the same model and fine-tune it with the same training iterations as the randomly initialized model. Results are shown in Table 4. It can be found that pre-training by CLAP also benefits LiDAR-based 3D object detection, even in a cross-dataset setting. And if we look at the performance of "Rand*" and "CLAP*", CLAP also accelerates the convergence in downstream task.

B. Visualization of Prototype Learning

We use the model pre-trained by CLAP to infer the 3D features and assign prototypes to different LiDAR points in the 3D space. Then we use different colors to indicate different prototypes and visualize them in Figure 5. If can be found that the background road plane inside the same frame is generally assigned to the same prototype. And foreground vehicles are assigned to another prototype. This demonstrate that the proposed prototype learning scheme actually learns some level of semantics understanding in an unsupervised manner. However, as our pre-training does not incorporate any label, it can also be found that the prototype assignment has some noise, for example some of the road plane points are assigned to other prototypes.

Init.	F.T.	mAP	Vehicle			Pedestrian			Cyclist		
			0-30m	30-50m	50m-	0-30m	30-50m	50m-	0-30m	30-50m	50m-
Rand*	5%	20.48	58.03	25.22	12.98	11.62	9.75	6.97	21.55	6.83	3.11
CLAP*		22.86 $^{+2.38}$	58.37	26.38	14.07	12.60	9.50	7.88	30.08	10.50	5.39
Rand		46.07	76.71	51.15	31.84	37.53	20.12	9.84	62.00	42.61	24.18
CLAP		46.88 $^{+0.81}$	76.98	51.64	31.31	38.79	20.60	9.74	63.75	43.21	26.83
Rand*	100%	64.00	86.21	70.20	58.20	57.80	41.18	23.55	75.95	61.45	45.80
CLAP*		64.74 $^{+0.74}$	88.14	72.59	59.13	57.37	42.24	24.22	77.11	61.91	45.63
Rand		65.03	88.18	74.23	61.75	57.32	38.90	21.96	78.07	64.32	48.16
CLAP		65.56 $^{+0.53}$	87.97	72.77	62.11	58.33	40.11	21.29	78.63	64.70	47.27

Table 4. Results for transferring experiments on Once [29] dataset. CenterPoint [54] is used as the downstream detector. "Init." indicates the initialization methodshorthands. "F.T." indicates the number of training samples in fine-tuning stage. Overall mAP and APs for different categories within different ranges are shown in this table. "Rand*" means training the randomly initialized model with the original training iterations in OpenPCDet [42]. "Rand" indicates that we increase the number of training iterations for randomly initialized model until convergence is observed. "CLAP*" indicates that we pre-train the backbones with CLAP on NuScenes [2] and then fine-tune on Once with the original iterations in [42]. "CLAP" uses the same fine-tuning iterations as "Rand". We use green color to highlight the performance improvement brought by CLAP. All the results are in %.

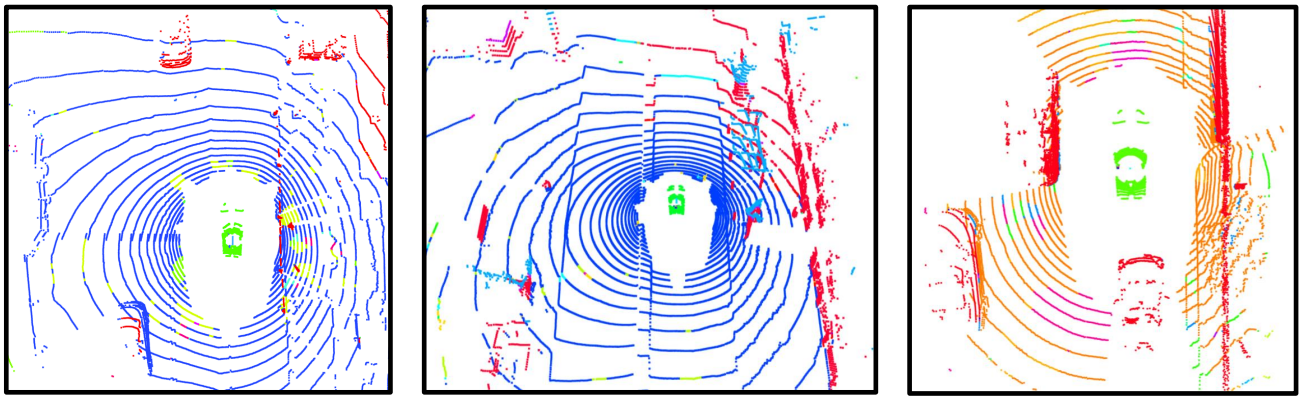

Figure 5. Visualization of the prototype learning results. Different color indicates different prototype assignments.