Geometrically-driven Aggregation for Zero-shot 3D Point Cloud Understanding

Guofeng Mei   Luigi Riz   Yiming Wang   Fabio Poiesi
Fondazione Bruno Kessler, Via Sommarive, 18, 38123 Trento, Italy
{gmei,luriz, ywang, poiesi}@fbk.eu

Abstract

Zero-shot 3D point cloud understanding can be achieved via 2D Vision-Language Models (VLMs). Existing strategies directly map VLM representations from 2D pixels of rendered or captured views to 3D points, overlooking the inherent and expressible point cloud geometric structure. Geometrically similar or close regions can be exploited for bolstering point cloud understanding as they are likely to share semantic information. To this end, we introduce the first training-free aggregation technique that leverages the point cloud's 3D geometric structure to improve the quality of the transferred VLM representations. Our approach operates iteratively, performing local-to-global aggregation based on geometric and semantic point-level reasoning. We benchmark our approach on three downstream tasks, including classification, part segmentation, and semantic segmentation, with a variety of datasets representing both synthetic/real-world, and indoor/outdoor scenarios. Our approach achieves new state-of-the-art results in all benchmarks. Code and dataset are available at https://luigiriz.github.io/geoze-website/.

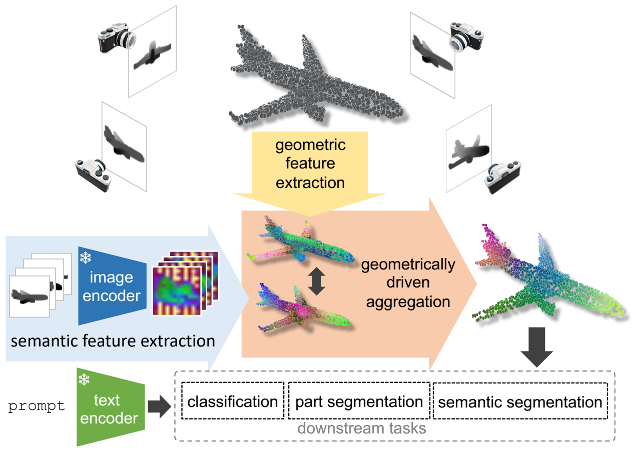


Figure 1. Given a set of dense per-pixel VLM representations (e.g. CLIP [22]) extracted from different viewpoint images of a point cloud, our approach is the first geometrically-driven aggregation technique to effectively transfer these image representations to 3D points. We use geometric features (e.g. FPFH [23], normals) extracted from the point cloud to denoise the VLM representations through an iterative process. This process begins by aggregating information locally and then extends to operate globally, thereby facilitating improvements across a variety of downstream tasks.

1. Introduction

Zero-shot point cloud understanding aims to develop deep learning models capable of performing recognition tasks, such as shape classification or segmentation, on data unobserved at training time [8, 18, 35]. Especially, vision-language models (VLMs), such as CLIP [22], trained on internet-scale image-text pairs, can be effectively employed for zero-shot visual recognition tasks [11]. However, training a 3D deep model using CLIP's dataset size and procedure is currently infeasible since point cloud-text pair data of the same scale as that used for CLIP is not yet available.

Transferring VLM representations from images to point clouds has proven to be the most effective strategy so far for achieving zero-shot point cloud understanding [8, 30, 33, 35]. For example, PointCLIP [33] pioneers a training-free approach to directly transfer and combine per-pixel VLM representations of multiple rendered views of a point cloud for zero-shot classification and part segmentation. Its successor PointCLIPv2 [35] enhances 3D understanding by refining view rendering quality. ConceptFusion [8] fuses per-object VLM representations into per-pixel representations, transferring them to 3D points without requiring training. Both PointCLIPv2 and ConceptFusion have been tested in downstream tasks involving 3D scene understanding. However, 3D points remain isolated when transferring features from 2D domain to 3D domain, lacking any bridge for intra-information flow. Additionally, current fusion operations often disregard the global context and spatial geometric structures. We argue that the geometric structure of a point cloud contains valuable information that could be used to enhance the quality of the transferred VLM representations. In practical terms, we can rely on the key assumption: VLM representations should exhibit local smoothness and global consistency when their geometric structures are similar. To the best of our knowledge, no existing training-free method has harnessed the geometric information when transferring VLM representations from pixels to 3D points.

In this paper, we introduce the first **geo**metrically-driven training-free approach to aggregate point-level VLM representations for **ze**ro-shot point cloud understanding (GeoZe). Unlike previous methods that employ naive pooling operations to transfer and aggregate VLM representations from images to 3D points [8, 18, 35], GeoZe harnesses both local and global structural information to enable geometric consistency of VLM representations across the point cloud. GeoZe leverages superpoints to aggregate local information from neighboring points and facilitates a global exchange among superpoints with similar geometric structures, promoting accuracy and computational efficiency for downstream tasks. We utilize geometric representations (3D descriptors [23]) to identify the structures sharing similar VLM representations. Our approach employs a weighted linear combination for aggregation, where the weights are calculated by jointly considering point cloud VLM representations, geometric representations, and coordinates. A critical aspect of our method is maintaining the original alignment of visual representations in the representation space to ensure compatibility with their corresponding language representations. To achieve this, we introduce the concept of VLM representation anchors. These anchors serve to correct potential offsets that may arise during the aggregation process, thereby preserving the integrity of the original representations. We evaluate GeoZe on three zero-shot downstream tasks (shape classification, part segmentation, and semantic segmentation) across five datasets (ModelNet40 [29], ObjectScanNN [25], ShapeNet-Part [31], ScanNet [5], and nuScenes [1]). We use various 3D data benchmarks, including synthetic/real-world datasets, indoor/outdoor settings, and LiDAR/RGBD acquisition sensors. GeoZe consistently outperforms the considered baseline methods by a significant margin in a total of nine experiments. In summary, our contributions are:

- We introduce GeoZe, the first geometrically-driven aggregation approach for zero-shot point cloud understanding, leveraging VLMs.
- We show GeoZe's versatility, being training-free and adaptable to various architectures for different downstream tasks.
- We establish new state-of-the-art benchmarks in zero-shot downstream tasks, including object classification, part segmentation, and semantic segmentation.

2. Related works

Zero-shot PCD understanding. The advent of large-scale VLMs has prompted a surge in zero-shot point cloud understanding [3, 4], including shape classification [6, 7, 30, 33], and dense semantic segmentation [8, 18, 32, 35]. Zero-shot point cloud understanding can be performed by training a 3D encoder tasked with aligning 3D representations with VLM representations projected from images [6, 7, 18, 30, 32, 35]. For example, ULIP [30] trains a 3D encoder to align 3D features within the same latent space as the multi-view visual features extracted by CLIP's visual encoder and the textual features processed by CLIP's text encoder. CG3D [6] also trains a 3D encoder and introduces prompt tuning for the visual encoder. CLIP2Point [7] retrains CLIP's original image encoder with multi-view depth maps using a contrastive approach. CLIP^2 [32] dissects 3D scenes into text-image-point cloud proxies and applies cross-modal pre-training to generate point-level VLM representations. OpenScene [18] distills multi-view pixel-level VLM representations into a 3D encoder to learn point-level representations. Zero-shot point cloud understanding can also be achieved in a training-free manner [8, 33, 35]. PointCLIP [33] renders multi-view depth maps from point clouds and aggregates view-level zero-shot predictions for 2D-to-3D knowledge transfer. PointCLIPv2 [35] incorporates large language models for 3D-specific text prompts and introduces dense depth rendering with a novel 2D-3D mapping, enabling dense point-level representation and, consequently, part segmentation tasks. ConceptFusion [8] seeks dense point-level representation for semantic scene understanding by exploiting off-the-shelf image segmenters to combine local crops and global representations from images to extract pixel-level features. All the aforementioned methods do not take geometrical attributes into account during the semantic feature aggregation process. In contrast, GeoZe incorporates geometric information during the VLM representation aggregation phase on the point cloud to enhance zero-shot performance across various downstream tasks. GeoZe does not require any learnable parameters and can be seamlessly integrated into any training-free method.

Point representation aggregation. Most methods learn to encode the geometry of point clouds via neural networks and supervised learning. PointNet [20] uses max pooling for aggregation, while PointNet++ [21] hierarchically merges representations at multiple scales. PointCNN [12] hierarchically aggregates local representations using convolution operators on a transformed feature space. RS-CNN [13] introduces ad-hoc convolutions that consider local neighborhood relations and geometric structure for aggregation. SPG [10] models the input point clouds as hierarchical super-points graphs, in which geometrical feature aggregation is performed with graph analysis. DGCNN [26] learns a graph for each point cloud and applies convolution operations on its edges. Point Transformer [34] introduces a transformer block to aggregate features via self-attention. Such operation can be computationally expensive, works like Fast Point Transformer [17] and Point TransformerV2 [28] mitigate this drawback. Instead

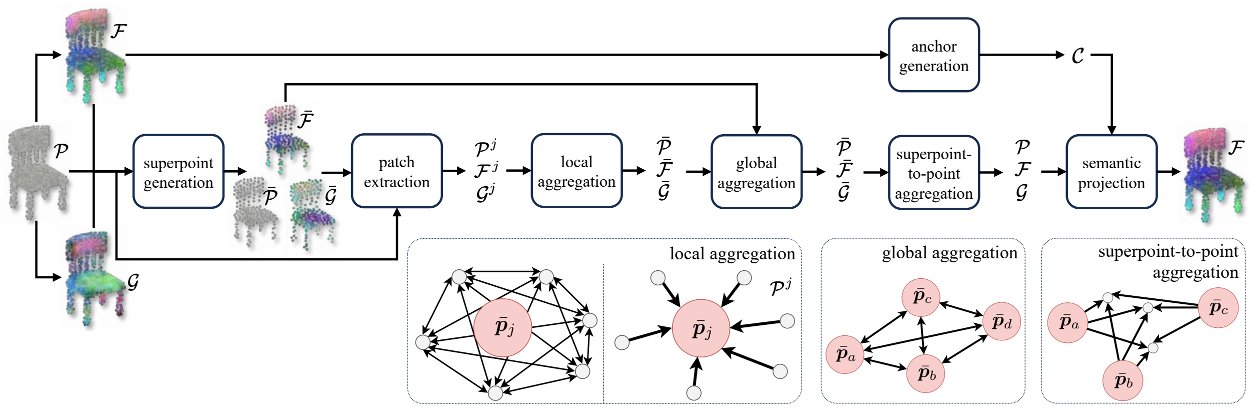

$\mathcal{F}$
$\mathcal{P}$
$\mathcal{G}$
superpoint generation
$\bar{\mathcal{F}}$
$\bar{\mathcal{P}}$
$\bar{\mathcal{G}}$
patch extraction
$\mathcal{P}^j$ $\mathcal{F}^j$ $\mathcal{G}^j$
local aggregation
$\bar{\mathcal{P}}$ $\bar{\mathcal{F}}$ $\bar{\mathcal{G}}$
global aggregation
$\bar{\mathcal{P}}$ $\bar{\mathcal{F}}$ $\bar{\mathcal{G}}$
superpoint-to-point aggregation
$\mathcal{P}$ $\mathcal{F}$ $\mathcal{G}$
anchor generation
$\mathcal{C}$
semantic projection
$\mathcal{F}$
local aggregation
$\bar{p}_j$
$\mathcal{P}^j$
global aggregation
$\bar{p}_a$ $\bar{p}_b$ $\bar{p}_c$ $\bar{p}_d$
superpoint-to-point aggregation
$\bar{p}_a$ $\bar{p}_b$ $\bar{p}_c$

Figure 2. Overview of the GeoZe framework. GeoZe first clusters point cloud $\mathcal{P}$ into superpoints $\bar{\mathcal{P}}$ along with their associated geometric representation $\bar{\mathcal{G}}$, VLM representation $\bar{\mathcal{F}}$, and anchors $\mathcal{C}$. For each superpoint $\bar{p}_j$, we identify its kNN within the point cloud to form a patch $\mathcal{P}^j$ with their features $\mathcal{G}^j$ and $\mathcal{F}^j$. For each patch, we perform a local feature aggregation to refine the VLM representations $\mathcal{F}$. The superpoints then undergo a process of global aggregation. A global-to-local aggregation process is applied to update the per-point features. Lastly, we employ the VLM feature anchors to further refine per-point features, which are then ready to be utilized for downstream tasks.

of learning semantic representations directly from 3D geometry, one can exploit multi-view rendering to learn 3D representation with additional visual inputs [2, 16, 27]. Recent zero-shot point cloud understanding methods transfer VLM representation to 3D either via learning a 3D encoder with backbones from PointNet++ [21] or PointMLP [14], or by mapping dense multi-view visual representation onto the 3D point cloud without training [8, 35]. OpenScene [18] projects multi-view pixel-level VLM representations on the 3D points, and multiple VLM representations for a point are fused via average pooling. ConceptFusion [8] employs SLAM techniques to incrementally integrate pixel-level VLM representations into the 3D space with a momentum-based update scheme. Also PointCLIPv2 [35] uses average pooling to fuse multi-view pixel-level VLM representations. The aforementioned aggregation methods either consider 3D geometry employing learning-based methods or simply aggregate VLM representations without taking into account local and global geometrical structure. Differently, GeoZe can effectively aggregate VLM representation in a geometrically-driven manner without training.

3. Our approach

3.1. Overview

We begin by extracting dense pixel-level VLM representations, e.g. with CLIP [22], from each viewpoint image of a point cloud. The image can be rendered [35] or be the original one used to reconstruct the point cloud [18]. These representations are then projected onto the 3D points. We compute point-level geometric representations, such as Fast Point Feature Histograms (FPFH) [23] or normal, to guide the aggregation of our VLM representations. By utilizing point cloud coordinates and geometric representations, we generate superpoints along with their associated geometric and VLM representations. For each superpoint, we identify its kNN within the point cloud to form a patch. For each patch we perform a local feature aggregation to denoise the VLM representations. Then, superpoints undergo a process of global aggregation. We apply Mean Shift clustering on point-level VLM representations to establish VLM representation anchors. A global-to-local aggregation process is applied to update the point representations, based on the coordinate distance and VLM representations similarity between points and superpoints. Lastly, we employ the VLM representation anchors to further refine point-level representations, which become then ready to be utilized for downstream tasks. The diagram of GeoZe is shown in Fig. 2.

3.2. Problem formulation

Let $\mathcal{P} = \{\boldsymbol{p}_i \in \mathbb{R}^3\}_{i=1}^N$ be a point cloud composed of N 3D points $\boldsymbol{p}_i$. For each $\boldsymbol{p}_i$ we compute and associate a VLM representation $\boldsymbol{f}_i \in \mathbb{R}^b$, where b is the dimension of $\boldsymbol{f}_i$. Let $\mathcal{F} = \{\boldsymbol{f}_i \in \mathbb{R}^b\}_{i=1}^N$ be the set of VLM representations associated to each 3D point. For each point $\boldsymbol{p}_i$, we compute and associate a geometric representation $\boldsymbol{g}_i \in \mathbb{R}^d$, where d is the dimension of $\boldsymbol{g}_i$. Let $\mathcal{G} = \{\boldsymbol{g}_i \in \mathbb{R}^d\}_{i=1}^N$ be the set of geometric representations associated to each 3D point. GeoZe transforms $\mathcal{F}$ in geometrically-coherent VLM representations by combining the information from $\mathcal{P}$, $\mathcal{F}$, and $\mathcal{G}$ based on local and global information through an iterative aggregation process. $\mathcal{F}$ can be used for downstream tasks, such as classification and segmentation.

3.3. VLM representation anchors and superpoints

The cardinality of point clouds is typically large, hence we compute superpoints and perform computations on them. A superpoint is a cluster of points featuring homogeneous geometric and VLM representations within a neighborhood of points. Information can be propagated back to individual points, resulting in dense point estimations. Intervening on VLM representations may lead to representations that diverge from the original feature space. Hence, we calculate a set of VLM representation anchors from the superpoints, which act as a reference for determining the final point-level representations. Steps for this computation are: i) initialize a set of point seeds uniformly distributed over the point cloud; ii) compute superpoints based on both geometric and visual information; iii) compute anchors based on aggregates of points that exhibit similar VLM representations.

Seed initialization. We compute seeds through Farthest Point Sampling (FPS) [21]. Specifically, we apply FPS on $\boldsymbol{\mathcal{P}}$ to sample $\bar{N}$ seeds based on point coordinates. Let $\bar{\boldsymbol{\mathcal{P}}} = \{\bar{\boldsymbol{p}}_j \in \mathbb{R}^3\}_{j=1}^{\bar{N}}$ be the set of seeds. The corresponding VLM representations and geometric features are $\bar{\boldsymbol{\mathcal{F}}} = \{\bar{\boldsymbol{f}}_j \in \mathbb{R}^b\}_{j=1}^{\bar{N}}$, and $\bar{\boldsymbol{\mathcal{G}}} = \{\bar{\boldsymbol{g}}_j \in \mathbb{R}^d\}_{j=1}^{\bar{N}}$, respectively.

Superpoint computation. Points proximate in the representation space are not necessarily proximate in the coordinate space. We aim to produce superpoints that uniformly partition the entire point cloud; thus, we formulate this as an optimal transport problem [15, 19]. Specifically, we compute superpoints iteratively by searching for centroids in the coordinate, geometric, and VLM representation spaces. This is achieved by initializing and updating their respective seeds, $\bar{\boldsymbol{\mathcal{P}}}$, $\bar{\boldsymbol{\mathcal{G}}}$, and $\bar{\boldsymbol{\mathcal{F}}}$. Firstly, for a given j-th seed, we select a set of K_1 nearest neighbors, denoted as $\mathcal{N}_j$, in the coordinate space with respect to $\bar{\boldsymbol{p}}_j$, and compute the average similarity $\bar{\boldsymbol{\mu}} = \{\bar{\boldsymbol{\mu}}_j\}$ with $\bar{\boldsymbol{\mu}}_j = \frac{1}{2K_1}\sum_{k=1}^{K_1}\left(1.0 + \frac{\bar{\boldsymbol{f}}_j^k \bar{\boldsymbol{f}}_j}{\|\bar{\boldsymbol{f}}_j^k\|\|\bar{\boldsymbol{f}}_j\|}\right)$. Then, we solve the following optimization problem:

$$\min_{\boldsymbol{\gamma}} \sum_{ij} \boldsymbol{\gamma}_{ij}\left(\frac{1}{\sqrt{D_c}}\|\boldsymbol{p}_i - \bar{\boldsymbol{p}}_j\|_2 + \frac{1}{\sqrt{D_g}}\|\boldsymbol{g}_i - \bar{\boldsymbol{g}}_j\|_2\right), \quad (1)$$
$$\sum_i \boldsymbol{\gamma}_{ij} = \bar{\boldsymbol{\mu}}_j / \sum_k \bar{\boldsymbol{\mu}}_j, \quad \sum_j \boldsymbol{\gamma}_{ij} = 1/N,$$

where $\boldsymbol{\gamma}_{ij}$ is the probability of affiliation of $\boldsymbol{p}_i$ to $\bar{\boldsymbol{p}}_j$, and $\boldsymbol{\gamma} = \{\boldsymbol{\gamma}_{ij}\}_{ij}^{N\bar{N}}$ is the set of these probabilities. D_c and D_g represent the average of the shortest distances between superpoints in the coordinate and geometric spaces, respectively. Eq. (1) can be solved by using the Sinkhorn algorithm [19]. We update the elements of $\bar{\boldsymbol{\mathcal{P}}}$, $\bar{\boldsymbol{\mathcal{G}}}$ and $\bar{\boldsymbol{\mathcal{F}}}$ as

$$\bar{\boldsymbol{p}}_j \leftarrow \frac{\sum_{i\in\mathcal{N}_i}\boldsymbol{\gamma}_{ij}\boldsymbol{p}_i}{\sum_{i\in\mathcal{N}_i}\boldsymbol{\gamma}_{ij}},\ \bar{\boldsymbol{g}}_j \leftarrow \frac{\sum_{i\in\mathcal{N}_i}\boldsymbol{\gamma}_{ij}\boldsymbol{g}_i}{\sum_{i\in\mathcal{N}_i}\boldsymbol{\gamma}_{ij}},\ \bar{\boldsymbol{f}}_j \leftarrow \frac{\sum_{i\in\mathcal{N}_i}\boldsymbol{\gamma}_{ij}\boldsymbol{f}_i}{\sum_{i\in\mathcal{N}_i}\boldsymbol{\gamma}_{ij}}.$$

This formulation prevents superpoints that are connected in the feature space from being grouped into the same categories if they are disjoint in 3D space. We alternately iterate $\bar{\boldsymbol{\mu}}$, $\bar{\boldsymbol{\mathcal{P}}}$, $\bar{\boldsymbol{\mathcal{G}}}$ and Eq. (1) Γ times to converge to a stable solution.

VLM representation anchors. We model anchors as centroids of clusters characterized by homogeneous VLM representations and employ Mean-Shift for iterative cluster determination [9]. We use Mean-Shift over other clustering methodologies, such as KMeans or Mixture Models, due to its ability to adaptively vary the number of clusters based on a kernel bandwidth and to be density aware. This approach can be particularly effective because the clusters we seek are those characterized by high-density points with distinctive representations, allowing each anchor centroid to potentially represent a semantic category. Clusters with lower densities can instead be more likely to be representative of noisy points. We introduce an optimized version of the Mean Shift clustering algorithm by considering both geometric and vision features. This is implemented by initializing the anchors with the original representations of the seeds, $\boldsymbol{\mathcal{C}}_v^0 = \{\boldsymbol{\mathcal{C}}_{v_j}^0 = \bar{\boldsymbol{f}}_j\}_{j=1}^{\bar{N}}$ and $\boldsymbol{\mathcal{C}}_g^0 = \{\boldsymbol{\mathcal{C}}_{g_j}^0 = \bar{\boldsymbol{g}}_j\}_{j=1}^{\bar{N}}$. Mean-Shift is iterative and the t-th iteration is computed as:

$$\boldsymbol{\mathcal{C}}_s^{t+1} \leftarrow \boldsymbol{Z}\boldsymbol{K}^t\boldsymbol{D}^{-1}, \boldsymbol{D} \leftarrow \text{diag}(\boldsymbol{K}^{t\top}\mathbf{1}),$$
$$\boldsymbol{K}^t \leftarrow \exp\left(\frac{\boldsymbol{\mathcal{F}}\boldsymbol{\mathcal{C}}_v^{t\top}}{\delta_v^2\|\boldsymbol{\mathcal{F}}\|\|\boldsymbol{\mathcal{C}}_v\|}\right)\exp\left(\frac{\boldsymbol{\mathcal{G}}\boldsymbol{\mathcal{C}}_g^{t\top}}{\delta_g^2\|\boldsymbol{\mathcal{G}}\|\|\boldsymbol{\mathcal{C}}_g\|}\right), \quad (2)$$

where $\delta_s(s{\in}\{v,g\})$ is the bandwidth that corresponds to the average similarity from each point to its 16-th nearest neighbor within the VLM representation space, and $\|\cdot\|$ is the L_2 norm. Mean-shift iterations persist until convergence, which in our experiments typically occurs around 40 iterations. Centroids are then computed using non-maximum suppression: starting with the point of maximum density, all proximates with a similarity greater than the threshold $\delta_s/2(s{\in}\{v,g\})$ are excluded, with the process iteratively repeated. $\delta_s(s{\in}\{v,g\})$ is the bandwidth that corresponds to the average similarity from each centroid to its 16-th nearest neighbor within the feature space. Points are subsequently allocated to segments, according to proximity to their closest cluster center. We iterate this procedure about 40 times to reach the final count of centroids L, which serve as the label prototypes $\boldsymbol{\mathcal{C}}_v, \boldsymbol{\mathcal{C}}_g$ for subsequent analyses.

3.4. Geometrically-driven aggregation

Given the superpoints and VLM representation anchors, we perform an aggregation of VLM representations at point level. We aggregate VLM representations locally within the neighborhood of each superpoint and subsequently conduct a global aggregation among the superpoints. This information is aggregated from superpoints to individual points, assigning a VLM representation to each point.

Local aggregation. Assuming that VLM representations should exhibit local smoothness, we perform local aggregation through a linear combination of VLM representations in the neighborhood of each superpoint. Specifically, for the j-th superpoint, we select a set of K_2 nearest-neighbor points defined as $\boldsymbol{\mathcal{P}}^j$ for the point coordinates, $\boldsymbol{\mathcal{F}}^j$ for the point-level VLM representations, and $\boldsymbol{\mathcal{G}}^j$ for the point-level geometric representations. We compute the linear combination weights as the solution to an optimal transport problem using the Sinkhorn algorithm [19]. We apply the Sinkhorn algorithm based on the similarity of points in terms of their geometric and VLM representations. This approach allows us to regulate the contributions from neighboring points, mitigating situations where a few highly similar points disproportionately influence the linear combination. We define the geometric similarity as $\boldsymbol{S}_g^j = \boldsymbol{\mathcal{G}}^j {\boldsymbol{\mathcal{G}}^j}^\top / \sqrt{d}$, and the VLM representation similarity as $\boldsymbol{S}_v^j = \boldsymbol{\mathcal{F}}^j {\boldsymbol{\mathcal{F}}^j}^\top / \sqrt{b}$. Therefore, we compute the linear combination weights as $\mathbf{W}_g^j = \mathtt{SH}\left(\boldsymbol{S}_g^j\right), \mathbf{W}_v^j = \mathtt{SH}\left(\boldsymbol{S}_v^j\right)$, where SH is the Sinkhorn function with 5 iterations. We combine, normalize, and use these weights to aggregate the VLM representations

$$\mathbf{W}^j = \mathtt{SM}(\mathbf{W}_g^j * \mathbf{W}_v^j), \quad \boldsymbol{\mathcal{F}}^j = \frac{1}{2}\left(\boldsymbol{\mathcal{F}}^j + \mathbf{W}^j \boldsymbol{\mathcal{F}}^j\right), \tag{3}$$

where SM is the Softmax operation. Then, we use the refined local features ($\boldsymbol{\mathcal{F}}^j$) to update the superpoint features by $\bar{\boldsymbol{\mathcal{F}}} = \gamma \bar{\boldsymbol{\mathcal{F}}}$. γ is same as in Eq. (1). Once all the superpoints are processed, the next step is the global aggregation.

Global aggregation. Geometrically similar regions may belong to similar objects, hence their VLM representations should be similar too. In order to aggregate VLM representations of objects with similar geometric features we expand our search at global level. As for the local aggregation, we use the Sinkhorn algorithm based on the similarity of geometric and VLM representations. Let $\bar{\boldsymbol{S}}^g$ be the geometric similarity computed among all superpoints, where each element is computed as $\bar{\boldsymbol{S}}_{ij}^g = \bar{\boldsymbol{g}}_i \bar{\boldsymbol{g}}_j^\top / \sqrt{d}$. Similarly, let $\boldsymbol{S}^v$ be the VLM representation similarity, where each element is computed as $\bar{\boldsymbol{S}}_{ij}^v = \bar{\boldsymbol{f}}_i \bar{\boldsymbol{f}}_j^\top / \sqrt{b}$. The weights $\bar{\boldsymbol{W}}$ for the global aggregation are expressed as:

$$\bar{\boldsymbol{W}}^g = \mathtt{SH}\left(\bar{\boldsymbol{S}}^g\right), \bar{\boldsymbol{W}}^v = \mathtt{SH}\left(\bar{\boldsymbol{S}}^v\right), \bar{\boldsymbol{W}} = \mathtt{SM}(\bar{\boldsymbol{W}}^g * \bar{\boldsymbol{W}}^v * \bar{\boldsymbol{M}}),$$

where $\bar{\boldsymbol{M}}$ is a mask matrix with their element $\bar{\boldsymbol{M}}_{ij} = 1$ if $\|\bar{\boldsymbol{p}}_i - \bar{\boldsymbol{p}}_j\|_2 < D_c$, otherwise $\bar{\boldsymbol{M}}_{ij} = 0$. Then, the superpoint level features are updated by $\bar{\boldsymbol{\mathcal{F}}} = \frac{1}{2}\left(\bar{\boldsymbol{\mathcal{F}}} + \bar{\boldsymbol{W}} \bar{\boldsymbol{\mathcal{F}}}\right)$.

Superpoint-to-point aggregation. We aggregate the updated superpoints' VLM representations together with their neighboring points. We use the distance between the point and each superpoint to weight the contribution of the VLM representation to aggregate. Specifically, given a query point $\boldsymbol{p}_i \in \boldsymbol{\mathcal{P}}$, the weight of superpoint $\bar{\boldsymbol{p}}_j \in \bar{\boldsymbol{\mathcal{P}}}$ to query point is decided by their coordinate distance $\boldsymbol{S}_{ij}^c = \mathtt{SM}\left(D_c - \|\boldsymbol{p}_i - \bar{\boldsymbol{p}}_j\|_2\right)$ and feature similarity $\boldsymbol{S}_{ij}^v = \boldsymbol{f}_i \bar{\boldsymbol{f}}_j^\top / \sqrt{b}$. We opted for the tanh kernel due to its ability to sharpen the distance measure, thereby ensuring that points are primarily influenced by their closest superpoints in the coordinate space. Let $\boldsymbol{S}^c = \{\boldsymbol{S}_{ij}^c\}_{i,j=1,1}^{N,\bar{N}}$ and $\boldsymbol{S}^v = \{\boldsymbol{S}_{ij}^v\}_{i,j=1,1}^{N,\bar{N}}$. Then, the weight $\boldsymbol{W}$ is calculated by

$$\boldsymbol{W}^c = \mathtt{SH}\left(\boldsymbol{S}^c\right), \boldsymbol{W}^v = \mathtt{SH}\left(\boldsymbol{S}^v\right), \boldsymbol{W} = \mathtt{SM}(\boldsymbol{W}^c * \boldsymbol{W}^v). \tag{4}$$

The point-level feature is updated via $\boldsymbol{\mathcal{F}} = \frac{1}{2}\left(\boldsymbol{\mathcal{F}} + \mathbf{W} \bar{\boldsymbol{\mathcal{F}}}\right)$ to produce our final aggregated representations.

Anchor projection. Noisy representations may be incorporated during aggregation, thus causing offsets within the VLM representation space. Therefore, we project VLM representations onto our VLM representation anchors as $\boldsymbol{c}_i = \arg\max_{\boldsymbol{c}_j \in \boldsymbol{\mathcal{C}}} \left(\boldsymbol{f}_i^\top \boldsymbol{\mathcal{C}}_v (\boldsymbol{g}_i^\top \boldsymbol{\mathcal{C}}_g)\right)$ This makes the aggregated VLM representations to be more inclined to resemble their corresponding semantic representations.

4. Results

4.1. Experimental setup

GeoZe is implemented using PyTorch and Open3D libraries. Experiments are executed on an NVIDIA A40 48GB GPU. We use FPFH [23] as geometric features. Additional setting details are in the Supplementary Material.

Prompting. We use the text prompting techniques proposed in the comparison methods. Specifically, for PointCLIPv2 [35] we use the prompts obtained via "GPT Prompting" [35]. For OpenScene [18] and ConceptFusion [8], we use the "prompt engineering" proposed in [18].

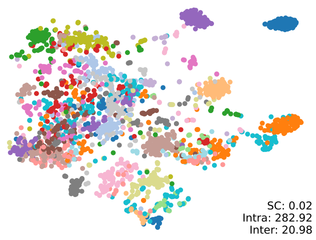
4.2. Shape Classification

Setting. We evaluate GeoZe on four datasets for shape classification tasks. Specifically, we use the synthetic dataset ModelNet40 [29], and three variants of the real-world dataset ObjectScanNN [25]. ModelNet40 [29] comprises 12,311 point clouds, with 2,468 of these designated for testing, uniformly sampled from CAD mesh surfaces across 40 categories. We report the results on the test set. ScanObjectNN [25] comprises 15 categories and 2,902 point clouds divided in: OBJ-ONLY (ground-truth segmented objects extracted from the scene beforehand, i.e., objects without background), OBJ-BG (objects attached with background data), and S-PB-T50-RS (objects perturbed with random rotation and scaling). We report the results computed on the test set that comprises 580 point clouds. GeoZe processes the depth maps extracted with PointCLIPv2 [35]. Unlike PointCLIPv2 that performs average pooling on the global representations of each depth map, we transform depth maps in point clouds, apply GeoZe, and then perform max pooling on the new VLM representations.

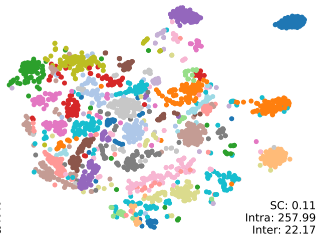
Results. Tab. 1 reports comparative results in terms of classification accuracy. GeoZe outperforms PointCLIPv2 on all datasets. Specifically, GeoZe achieves 70.17% on ModelNet40, outperforming PointCLIPv2 by +5.95. Moreover, GeoZe achieves 59.34%, 46.00%, and 39.87% on the three variations of ScanObjectNN. The margin with respect to PointCLIPv2 is significant, i.e., +18.03, +8.13, and +4.37 accuracy, respectively. In Fig. 3, we report a t-SNE comparison between the class representations extracted with PointCLIPv2 and GeoZe. We quantify t-SNE clusters with clustering metrics: Silhouette Coefficient, Inter-cluster Distance, and Intra-cluster Distance. The metrics underscore

Table 1. Zero-shot 3D classification results on ModelNet40 [24], and ScanObjectNN [25] in terms of accuracy. Bold font denotes best performance. Keys: MN40: ModelNet40, SOO: S-OBJ-ONLY, SOB: S-OBJ-BG, SPB: S-PB-T50-RS. repo.: reported from the paper. repr.: reproduced by us.

Method	MN40	SOO	SOB	SPB
PointCLIPv2 (repo.)	64.2	50.1	41.2	35.4
PointCLIPv2 (repr.)	63.3	41.3	37.9	35.5
GeoZe	**70.2**	**59.3**	**46.0**	**39.9**
Δ_{Accuracy}	+6.8	+18.0	+8.1	+4.4



(a) PointCLIPv2 [35]



(b) GeoZe

Figure 3. T-SNE embeddings of (a) PointCLIPv2 [35] and (b) GeoZe on ModelNet40. GeoZe produces better separated and grouped clusters for different categories, as evidenced by the superior silhouette coefficient (SC) and greater inter-cluster distance (inter), alongside a smaller intra-cluster distance (intra).

the efficacy of GeoZe in more effectively separating point features from diverse categories.

4.3. Part Segmentation

Setting. We evaluate GeoZe on ShapeNetPart [31] for the part segmentation task. We compare GeoZe against PointCLIPv2 [35]. ShapeNetPart includes 2,874 different objects, divided in 16 categories, and containing a total of 50 different part labels. These annotations are at point level. Following the evaluation procedure in PointCLIPv2 [35], we randomly sample 2,048 points from each point cloud.
Results. Tab. 2 reports the results in terms of the mean intersection of union (mIoU). For PointCLIPv2, we report the results presented in the original paper and the ones we reproduced by executing their released source code. GeoZe outperforms PointCLIPv2 by $+5.6$, achieving 57.4 mIoU. GeoZe improves the part segmentation quality for 14 out of 16 objects types, with very large margin in the case of *chair* ($+14.6$), *earphone* ($+7.2$), *laptop* ($+13.5$) and *mug* ($+15.2$). Fig. 4 reports qualitative part segmentation examples obtained with PointCLIPv2 and GeoZe. GeoZe can effectively denoise features, thus producing more homogeneous segmented parts. There are cases, like the aeroplane, that are difficult to improve due to the high level of noise.

4.4. Semantic Scene Segmentation

Setting. We evaluate GeoZe on ScanNet [5] and nuScenes [1] for the semantic scene segmentation task. ScanNet is an RGBD dataset including 2.5M views of 1,513 indoor scenes, annotated with point-level semantic labels and comprising 20 object classes. We evaluate methods on the original validation set. nuScenes is a LiDAR dataset of 400K outdoor scene point clouds with semantic annotations for 16 different classes. We apply GeoZe to the 3D point-level features extracted with different 2D feature extractors, specifically those used in OpenScene [18], based on OpenSeg and LSeg, and that used in ConceptFusion [8].
Results. In Tab. 3 we report the original results of OpenScene [18] and the ones reproduced by us with the *"2D Fusion"*-version of their approach in the indoor dataset ScanNet. *2D Fusion* is the only training-free version of OpenScene. For ConceptFusion [8], we reproduced the results with the original code[1]. GeoZe outperforms OpenScene (OpenSeg), OpenScene (LSeg), and ConceptFusion by $+2.0$, $+3.9$, and $+3.1$ mIoU on average, respectively. Specifically, GeoZe outperforms OpenScene (OpenSeg) on 17 out 20 classes of ScanNet, with significant gains on *wall* (+5.8), *floor* (+8.7), and *bathtub* (+6.2) classes. GeoZe outperforms OpenScene (LSeg) on all the 20 classes, in particular for *floor* (+7.5), *bathtub* (+6.2), and *toilet* (+9.6) classes. GeoZe outperforms ConceptFusion on 19 out of 20 classes, with a significant margin in the case of *bathtub* (+18.5), *books* (+5.6), and *floor* (+5.2) classes.

Tab. 4 reports the results in the outdoor dataset nuScenes. Similarly to Tab. 3, we reproduce the results of the *2D Fusion*-version of OpenScene [18] using the OpenSeg feature extractor as it is the only one available. GeoZe outperforms OpenScene (OpenSeg) by 2.0 mIoU on average, and on 11 out of the 16 classes, with a improvement on *bus* by 8.1 mIoU, *truck* by 6.3 mIoU, and *drivable surface* by 15.2 mIoU. Fig. 5 reports examples of qualitative results. GeoZe produces more homogeneous segmented regions and reduces noise, particularly at the point cloud boundaries.

4.5. Ablation study

We use the part segmentation task on ShapeNetPart for the ablation study of GeoZe. The Supplementary Material contains additional ablation studies.
Effectiveness of aggregation levels. We assess the contributions of the local, global, and superpoint-to-point aggregations, and superpoint computation and anchor projection steps: for convenience, we name these steps as L, G, SP, SC and A. Since GeoZe first clusters a point cloud into superpoints and then propagates them back to its original resolution, we adopt the propagation strategy outlined in [21] as our baseline, in short B. Fig. 6 reports the results of our ablation study. B scores 54.0 mIoU. Superpoint computation (SC) and superpoint-to-point aggregation (SP) improve the baseline by 0.7 and 1.0 mIoU, respectively. The combination SC+SP results in a gain of 1.7 mIoU. Local aggre-

[1]These results are different from those reported in the original paper [8] because authors only evaluated a subset of the classes and scenes.

Table 2. Zero-shot part segmentation results on ShapeNetPart [31] in terms of mean Intersection over Union (mIoU). GeoZe outperforms PointCLIPv2's [35] on 14 out of 16 classes. Bold denotes best performance. Key: repo.: reported from the paper. repr.: reproduced by us.

	mIoU	Airpl.	Bag	Cap	Car	Chair	Earph.	Guitar	Knife	Lamp	Laptop	Motor.	Mug	Pistol	Rocket	Skate	Table
PointCLIPv2 (repo.)	49.5	33.5	60.4	52.8	-	51.5	56.5	71.5	66.7	-	61.6	-	48.0	-	**49.6**	43.9	61.1
PointCLIPv2 (repr.)	51.8	33.5	60.4	52.9	27.2	51.5	56.5	71.5	76.7	44.7	61.5	**31.5**	48.0	46.1	**49.6**	43.9	61.1
GeoZe (ours)	**57.4**	**33.6**	**70.2**	**64.7**	**33.6**	**66.1**	**63.7**	**73.6**	**77.9**	**45.4**	**74.0**	30.5	**63.2**	**48.3**	47.3	**45.6**	**63.8**
Δ_{IoU}	+5.6	+0.1	+9.8	+11.8	+7.4	+14.6	+7.2	+2.1	+1.2	+0.7	+13.5	-1.0	+15.2	+2.5	-2.3	+1.7	+2.7

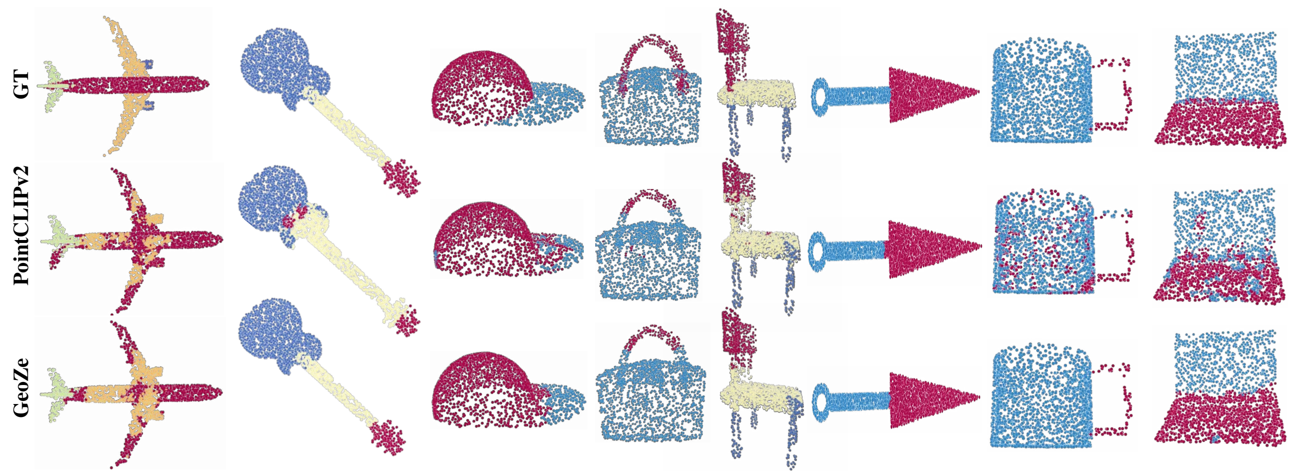


Figure 4. Zero-shot part segmentation results on ShapeNetPart [31]. (top row) ground-truth annotations, (middle row) PointCLIPv2 [35], and (bottom row) GeoZe. Parts segmented by GeoZe are more homogeneous than those segmented by PointCLIPv2.

Table 3. Zero-shot semantic segmentation results on ScanNet [5] in terms of mean Intersection over Union (mIoU). GeoZe outperforms the comparison methods (OpenScene [18] and ConceptFusion [8]) on all datasets. Bold font indicates best performance.

	mIoU	Wall	Floor	Cabin.	Bed	Chair	Sofa	Table	Door	Wind.	Books.	Pict.	Count.	Desk	Curt.	Refri.	Sh. C.	Toil.	Sink	Batht.	Oth. F.
OpenScene [18] (OpenSeg)																					
Reported	41.4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Reproduced	43.6	58.4	72.3	33.5	66.2	45.4	48.6	31.7	**40.1**	50.3	59.8	**23.7**	31.9	25.9	**63.3**	26.2	40.1	59.8	19.9	69.0	**6.1**
GeoZe (ours)	**45.6**	**64.2**	**81.0**	**33.7**	**68.8**	**49.0**	**50.6**	**35.6**	39.4	**53.4**	**62.7**	18.9	**35.1**	**28.3**	61.9	**28.8**	**42.0**	**63.9**	**20.4**	**75.2**	4.0
Δ_{IoU}	+2.0	+5.8	+8.7	+0.6	+2.6	+3.6	+1.6	+3.9	-0.7	+3.1	+2.9	-4.8	+3.2	+2.4	-2.4	+2.6	+1.9	+4.1	+0.5	+6.2	-2.1
OpenScene [18] (LSeg)																					
Reported	50.0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Reproduced	51.9	72.1	81.9	45.0	74.0	60.1	55.3	49.4	52.4	53.9	68.6	27.4	37.2	42.1	56.8	50.9	0.0	68.7	**47.9**	62.8	31.0
GeoZe (ours)	**55.8**	**75.5**	**88.4**	**47.3**	**78.1**	**67.9**	**58.3**	**54.5**	**53.3**	**59.4**	**73.6**	26.7	**44.6**	**46.5**	**57.0**	**56.0**	0.0	**75.5**	47.1	**71.4**	**33.6**
Δ_{IoU}	+3.9	+3.4	+7.5	+2.3	+4.1	+2.8	+3.0	+5.1	+0.9	+1.5	+5.0	-0.7	+7.4	+4.4	+0.2	+5.1	0.0	+9.6	-0.8	+8.6	+2.6
ConceptFusion [8]																					
Reproduced	23.4	17.9	38.6	22.4	44.7	31.8	33.8	24.9	36.5	20.1	36.7	8.4	2.6	9.8	30.7	13.5	19.1	13.8	17.3	39.8	5.8
GeoZe (ours)	**26.5**	**21.9**	**43.9**	**23.3**	**46.4**	**32.8**	**35.1**	**25.4**	**38.4**	**23.4**	**42.3**	8.0	**3.0**	**10.8**	**34.4**	**16.6**	**22.7**	**17.2**	**17.7**	**58.3**	**6.5**
Δ_{IoU}	+3.1	+4.0	+5.2	+0.9	+1.7	+0.9	+1.3	+0.5	+1.9	+3.3	+5.6	-0.4	+0.4	+1.0	+3.7	+3.1	+3.6	+3.4	+0.4	+18.5	+0.7

Table 4. Semantic segmentation results on nuScenes [1] in terms of mean Intersection over Union (mIoU), using OpenSeg feature extraction. GeoZe outperforms the OpenScene on 11 out of 16 classes. Bold font indicates best performance. Key: repr.: reproduced by us.

	mIoU	Barr.	Bicyc.	Bus	Car	Const.	Motor.	Pers.	Traf. c.	Trail.	Truck	Dr. Sur.	Oth. Fl.	Sidew.	Terr.	Mann.	Veget.
OpenScene (repr.)	20.6	1.9	**17.1**	31.1	28.6	14.8	**20.9**	**18.5**	5.1	13.9	19.6	34.7	0.3	**15.2**	23.3	**39.5**	45.8
GeoZe (ours)	**22.6**	**2.0**	16.3	**39.2**	**35.8**	**15.2**	18.7	12.9	**7.1**	**15.7**	**25.9**	**49.9**	**1.3**	14.9	**27.0**	33.2	**47.3**
Δ_{IoU}	+2.0	+0.1	-0.8	+8.1	+7.2	+0.4	-2.2	-5.6	+2.0	+1.8	+6.3	+15.2	+1.0	-0.3	+3.7	-6.3	+1.5

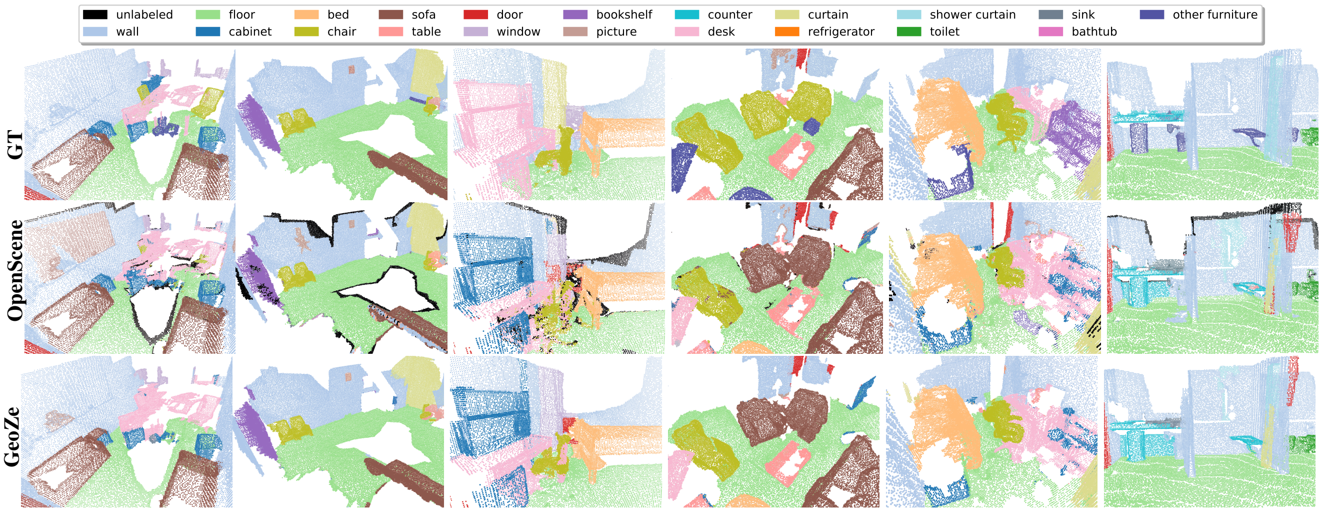


Figure 5. Zero-shot semantic segmentation results on ScanNet [5] using OpenSeg feature extraction (Tab. 3). (top row) ground-truth annotations, (middle row) OpenScene (OpenSeg), and (bottom row) GeoZe.

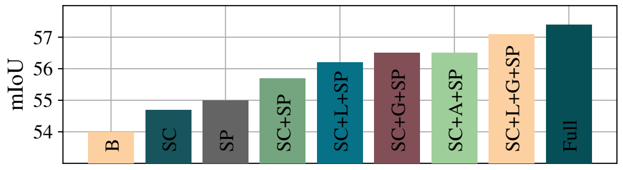


Figure 6. Ablation study on GeoZe's modules, tested on the part segmentation task. Keys: B: baseline. SC: superpoint computation. SP: superpoint-to-point aggregation. G: global aggregation. L: local aggregation. A: anchor projection. Full: the whole model.

gation (SC+L+SP) leads to an improvement of 0.5 mIoU over SC+SP, underlining the importance of local smoothing. Global aggregation (SC+G+SP) improves 0.8 mIoU over SC+SP, indicating the positive effect of global semantic context. Anchor projection (SC+A+SP) improves 0.8 mIoU over SC+SP, while the combination of SC+L+G+SP results in 57.1 mIoU. The full GeoZe reaches 57.4 mIoU.

Impact of feature types. Tab. 5 reports the results of the different types of features in the aggregation process. Our experiments indicate that incorporating geometric, semantic and coordinate information significantly enhances the effectiveness of local aggregation. Similarly, global aggregation based on geometric and semantic features allows for a great increase in performance. It improves the segmentation accuracy by nearly +1.2 mIoU (55.7% vs. 56.9%).

4.6. Computational analysis

We compare the processing time of GeoZe against the aggregation approaches of the evaluated methods and report the average time increment per point cloud. We run the baselines methods and GeoZe on a NVIDIA A40 48GB GPU and report inference time in the Tab. 6. Although we did not implement any low-level optimization of our algorithm, we can observe that the increment introduced by GeoZe on top of PointCLIPv2 and OpenScene is limited.

Table 5. Ablation study evaluating the influence of geometric information on part segmentation within the ShapeNetPart dataset. Keys: Coors: coordinate, CLIP: VLM representation, GA: global aggregation, LA: local aggregation.

	Feature	Feature Combinations							
LA	CLIP [35]	✓	✓	✓	✓				✓
	Coors		✓		✓				✓
	FPFH [23]			✓	✓				✓
GA	CLIP [35]					✓		✓	✓
	FPFH [23]						✓	✓	✓
	mIoU	55.7	56.2	56.2	56.4	55.7	56.5	56.9	57.4

Table 6. Comparison of inference times across different methods.

	ShapeNet		ScanNet	
	PointCLIPv2 [35]	GeoZe	OpenScene [18]	GeoZe
Time(ms)	7.52	9.83	2088.72	2125.61

5. Conclusions

We presented the first training-free aggregation technique that leverages the point cloud's 3D geometric structure to improve the quality of the transferred VLM representations. GeoZe is designed to iteratively aggregate representations at local level first and global level then. VLM representations are extracted from a CLIP model [22], while geometric representations are extracted from the FPFH algorithm [23]. We carried out an extensive evaluation on three downstream tasks: classification, part segmentation, and semantic segmentation. We reported the results on both synthetic/real-world, and indoor/outdoor datasets, and showed that GeoZe achieves new state-of-the-art results in all benchmarks.

Limitations. GeoZe depends on the quality of VLM representations, it may not accurately handle features of objects that constitute a small portion of the 3D scene, and it increases the inference time compared to a naive aggregation.

Acknowledgement. This work was supported by the PNRR project FAIR- Future AI Research (PE00000013), under the NRRP MUR program funded by the NextGenerationEU.

References

[1] 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 11621–11631, 2020. 2, 6, 7

[2] Xia Chen, Jianren Wang, David Held, and Martial Hebert. Panonet3d: Combining semantic and geometric understanding for lidar point cloud detection. In *3DV*, 2020. 3

[3] Ali Cheraghian, Shafin Rahman, Dylan Campbell, and Lars Petersson. Mitigating the hubness problem for zero-shot learning of 3d objects. In *BMVC*, 2019. 2

[4] Ali Cheraghian, Shafin Rahman, Townim F Chowdhury, Dylan Campbell, and Lars Petersson. Zero-shot learning on 3d point cloud objects and beyond. *IJCV*, 2022. 2

[5] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017. 2, 6, 7, 8

[6] Deepti Hegde, Jeya Maria Jose Valanarasu, and Vishal Patel. Clip goes 3d: Leveraging prompt tuning for language grounded 3d recognition. In *ICCV*, 2023. 2

[7] Tianyu Huang, Bowen Dong, Yunhan Yang, Xiaoshui Huang, Rynson WH Lau, Wanli Ouyang, and Wangmeng Zuo. Clip2point: Transfer clip to point cloud classification with image-depth pre-training. In *ICCV*, 2023. 2

[8] Krishna Murthy Jatavallabhula, Alihusein Kuwajerwala, Qiao Gu, Mohd Omama, Tao Chen, Shuang Li, Ganesh Iyer, Soroush Saryazdi, Nikhil Keetha, Ayush Tewari, Joshua B. Tenenbaum, Celso Miguel de Melo, Madhava Krishna, Liam Paull, Florian Shkurti, and Antonio Torralba. ConceptFusion: Open-set Multimodal 3D Mapping. In *RSS*, 2023. 1, 2, 3, 5, 6, 7

[9] Shu Kong and Charless C Fowlkes. Recurrent pixel embedding for instance grouping. In *CVPR*, 2018. 4

[10] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *CVPR*, 2018. 2

[11] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *ICML*, pages 19730–19742. PMLR, 2023. 1

[12] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. *NeurIPS*, 2018. 2

[13] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *CVPR*, 2019. 2

[14] Xu Ma, Can Qin, Haoxuan You, Haoxi Ran, and Yun Fu. Rethinking network design and local geometry in point cloud: A simple residual mlp framework. In *ICLR*, 2022. 3

[15] Guofeng Mei, Hao Tang, Xiaoshui Huang, Weijie Wang, Juan Liu, Jian Zhang, Luc Van Gool, and Qiang Wu. Unsupervised deep probabilistic approach for partial point cloud registration. In *CVPR*, pages 13611–13620, 2023. 4

[16] Seyed Saber Mohammadi, Yiming Wang, and Alessio Del Bue. Pointview-gcn: 3d shape classification with multi-view point clouds. In *ICIP*, 2021. 3

[17] Chunghyun Park, Yoonwoo Jeong, Minsu Cho, and Jaesik Park. Fast point transformer. In *CVPR*, 2022. 2

[18] Songyou Peng, Kyle Genova, Chiyu Jiang, Andrea Tagliasacchi, Marc Pollefeys, Thomas Funkhouser, et al. Openscene: 3d scene understanding with open vocabularies. In *CVPR*, 2023. 1, 2, 3, 5, 6, 7, 8

[19] Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport: With applications to data science. *Found. Trends Mach. Learn.*, 2019. 4

[20] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 2

[21] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *NeurIPS*, 2017. 2, 3, 4, 6

[22] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 1, 3, 8

[23] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *ICRA*, 2009. 1, 2, 3, 5, 8

[24] Abhishek Sharma, Oliver Grau, and Mario Fritz. Vconv-dae: Deep volumetric shape learning without object labels. In *ECCV*, 2016. 6

[25] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *ICCV*, 2019. 2, 5, 6

[26] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM TOG*, 2019. 2

[27] Xin Wei, Ruixuan Yu, and Jian Sun. View-gcn: View-based graph convolutional network for 3d shape analysis. In *CVPR*, 2020. 3

[28] Xiaoyang Wu, Yixing Lao, Li Jiang, Xihui Liu, and Hengshuang Zhao. Point transformer v2: Grouped vector attention and partition-based pooling. *NeurIPS*, 2022. 2

[29] 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. 2, 5

[30] Le Xue, Mingfei Gao, Chen Xing, Roberto Martín-Martín, Jiajun Wu, Caiming Xiong, Ran Xu, Juan Carlos Niebles, and Silvio Savarese. Ulip: Learning a unified representation of language, images, and point clouds for 3d understanding. In *CVPR*, 2023. 1, 2

[31] Li Yi, Vladimir G Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM TOG*, 2016. 2, 6, 7

[32] Yihan Zeng, Chenhan Jiang, Jiageng Mao, Jianhua Han, Chaoqiang Ye, Qingqiu Huang, Dit-Yan Yeung, Zhen Yang, Xiaodan Liang, and Hang Xu. Clip2: Contrastive language-image-point pretraining from real-world point cloud data. In *CVPR*, 2023. 2

[33] Renrui Zhang, Ziyu Guo, Wei Zhang, Kunchang Li, Xupeng Miao, Bin Cui, Yu Qiao, Peng Gao, and Hongsheng Li. Pointclip: Point cloud understanding by clip. In *CVPR*, 2022. 1, 2

[34] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *ICCV*, 2021. 2

[35] Xiangyang Zhu, Renrui Zhang, Bowei He, Ziyu Guo, Ziyao Zeng, Zipeng Qin, Shanghang Zhang, and Peng Gao. PointCLIPv2: Prompting CLIP and GPT for Powerful 3D Open-world Learning. In *ICCV*, 2023. 1, 2, 3, 5, 6, 7, 8